LIST

n°12

LE JOURNAL DES AMATEURS DE PROGRAMMATION

ISSN 0761-9936

SEPTEMBRE-OCTOBRE 1985

HISTOIRE DES LANGAGES

Les premiers programmes de l'humanité

BASIC ET MATHS

Vingt machines comparées

ASSEMBLEUR EN TROIS 7

TO7, TO7/70, et X-07

AMSTRAD

Jongler avec les couleurs

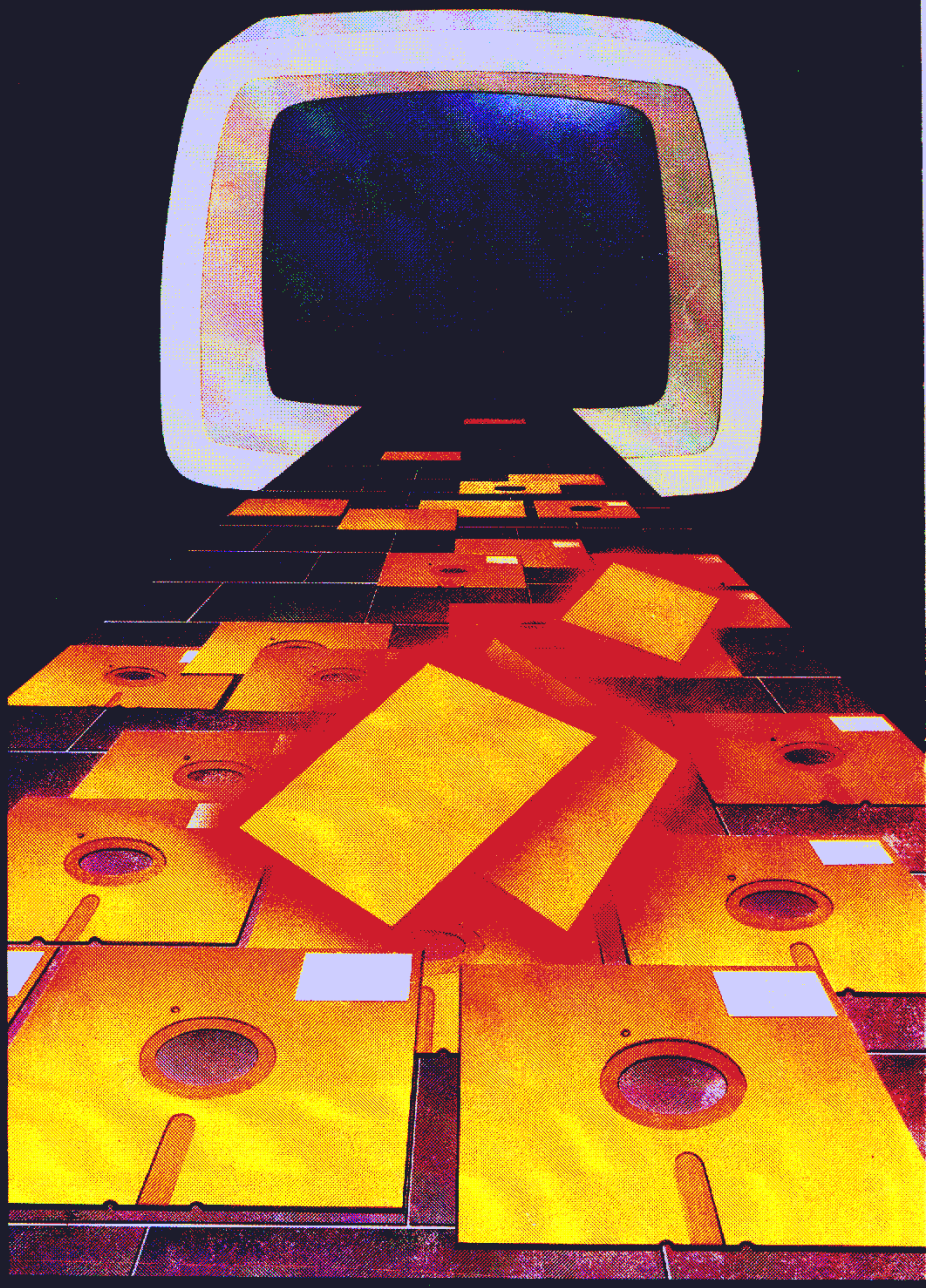
LES JEUX DE LIST

Des problèmes informatiques et leur solution

LA BOITE A MALICES

Des ficelles et des programmes

M 2712 - 12 - 20 F



Belgique : 154 FB - Canada : 2.95$C - Suisse : 6 FS

20 F

exelvi

LE DEFI MICRO INFOR

LE SYSTEME EXL 100

Simplicité, modularité, évolutivité sont les 3 mots-clés du Système EXL 100.

- 72 K octets
- Synthèse vocale intégrée
- Ecran à très haute définition graphique
- Prise Péritel incorporée
- Commandes infrarouges

UNE UNITE CENTRALE EXPLOSIVE
Une architecture bi-processeurs

- Les micro-processeurs: 2 processeurs 8 bits de la famille TMS 7000
- Le contrôleur de visualisation: exceptionnelle qualité graphique 80.000 points
- Le synthétiseur de parole: l'EXL 100 est équipé d'un synthétiseur de parole de très haute qualité: le TMS 5220 C
- Les émetteurs à infrarouge: la commande sans fils, par infrarouges du clavier et des manettes de jeu
- La mémoire statique
- Le prédiffusé
- L'alimentation intégrée

2.690 F TTC

Le junior: pour faire ses gammes
Ce clavier est conçu pour offrir, sous un encombrement très réduit, toutes les fonctions d'un clavier professionnel. Il est particulièrement étudié pour les mains d'enfants et pour pouvoir s'intégrer dans un environnement familial.
La commande de l'ordinateur EXL 100 par liaison infrarouge n'est pas un gadget. Elle permet d'utiliser l'EXL 100 sans fatigue pour les yeux, en se tenant à la distance idéale du téléviseur conseillée par son fabricant.

Le pro: pour le travail sérieux : 295 F TTC
Ce clavier est destiné aux fanas de la programmation et pour les utilisations type traitement de texte.

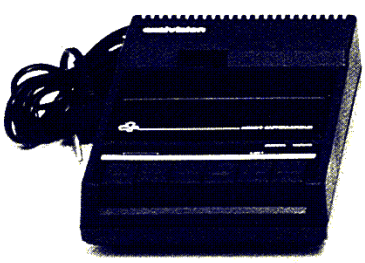
DES ACCESSOIRES ADAPTES A TOUTES LES UTILISATIONS

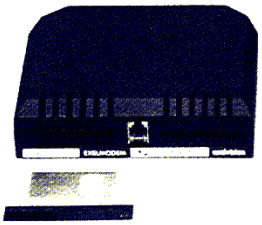
- Les cartouches ROM
- Le lecteur de cassettes (Data Recorder) **295 F TTC**
- Les manettes en option au prix de 250 F TTC
- EXELDRUMS (Boîte à rythmes)

DES ACCESSOIRES SUPER PROFESSIONNELS
L'imprimante EXL 80
C'est une petite merveille de technologie 100% française conçue et réalisé par CGCT et EURO-TERMINAL. Elle allie les plus hautes performances à une grande simplicité mécanique, donc à une grande robustesse. Silencieuse, économique, peu encombrante, c'est la championne de la copie d'écran. **3.500 F TTC**

Le moniteur monochrome
- Ecran vert 12 pouces à haute définition (Résolution moyenne: 900 lignes). Lecture agréable et sans fatigue.
- Connecteur Péritélévision incorporé.
- Encombrement limité (29x37x34 cm) faible poids (6,8 kg). Facilement transportable.

Exelmémoire (CMOS RAM): la mémoire-clé
Facilement transportable, beaucoup plus souple et rapide d'emploi qu'une cassette-audio, l'Exelmémoire peut constituer un véritable "cahier de classe" pour les élèves de l'âge de l'informatique, permettant de programmer chez soi, puis d'emmener son programme en classe, chez des amis. . . .





Elle peut également jouer le rôle d'une extension-mémoire de 16 Ko et constitue le complément indispensable de l'Exelmodem pour accéder aux fonctions de téléchargement de logiciels. **590 F TTC**

EXELVISION, UN SYSTEME OUVERT SUR LE MONDE
Exelmodem
Connecté à votre ligne téléphonique, l'Exelmodem permet de transformer l'EXL 100 en un véritable MINITEL-couleur. Il peut également être utilisé pour communiquer avec un autre EXL 100.
De plus, il possède les fonctions de numérotation automatique et de détection de sonnerie; en cas d'absence de votre part, un correspondant peut laisser un message écrit, donnant ainsi à l'EXL 100 une fonction de télex.

Avec Exelmodem et le réseau Antiope, l'ordinateur prend une autre dimension et vous donne accès à l'univers de la communication tel qu'il sera à la fin du siècle.
Avec Exelmodem, votre EXL 100 accède, par simple appel téléphonique, à toutes les structures d'information du système Vidéotex français créé pour le Minitel.

Le s
que
don
info
Ave
l'info
vous
Exe
(Un
stan
extr
calc
sur
com
les
virgu
Log
Cata
69 I

* Prix au 20.03.1985 valables dans la limite des stocks disponibles et sauf erreur d'impression.

vision
ORMATIQUE FRANCAIS

3250 F TTC*
Avec moniteur

***Configuration de base comprenant:**

- Une unité centrale de 72 K/Octets; 32 K RAM utiles extensibles à 48,72 K utiles et plus
- Un moniteur monochrome à très haute définition graphique
- Un clavier JUNIOR, 61 caractères, AZERTY. . . .
- Synthèse de la parole
- Prise Péritel incorporée
- Le logiciel EXELBASIC de 32 K ROM
- EN CADEAU:
  Cours d'initiation et d'autoformation au BASIC

DOM LISTE DES POINTS DE VENTE DOMICA

01000 BOURG EN BRESSE - DOMICA - 60, rue Charles Robin	Tél. (74) 22.42.77
01100 OYONNAX - DOMICA - 38, rue Brillat Savarin	Tél. (74) 73.62.55
07100 ANNONAY - DOMICA - 13, rue de Tournon	Tél. (75) 67.67.58
26000 VALENCE - DOMICA - 215, avenue Victor Hugo	Tél. (75) 41.14.75
3000 NIMES - DOMICA - 134, rue d'Avignon	Tél. (66) 27.28.29
33000 BORDEAUX - CIESO - 3, rue Capdeville	Tél. (56) 44.51.22
38000 GRENOBLE - DOM ALPES - 45, avenue Alsace Lorraine	Tél. (76) 87.16.26
38000 GRENOBLE - DOM ALPES - 6, rue Ampère	Tél. (76) 49.65.65
63000 CLERMONT FERRAND - DOMICA - 53, rue Bonnabaud	Tél. (73) 35.51.40
67000 STRASBOURG - DOM ALSACE - 5, rue des Frères	Tél. (88) 35.76.16
69002 LYON - DOM - 63, Passage de l'Argue	Tél. (7) 837.76.14
69007 LYON - DOM - 274, rue de Créqui	Tél. (7) 872.49.52
71100 CHALONS/SAONE-MICROCAL DOMICA-22,quai de la Poterne	Tél.(85)48.98.57
84000 AVIGNON DOMICA - 81 rue Bonneterie	Tél. (90) 85 53 14

Le système Antiope utilise les ondes hertziennes (les mêmes que celles de votre téléviseur et des liaisons satellites) dont les signaux sont décodés pour vous transmettre des informations.
Avec des 2 systèmes, Exelvision créé un lien direct entre l'informatique et les techniques de communication. . . . et vous ouvre le monde. **1.090 F TTC**

Exelbasic
(Un Basic de 32 K de ROM). Il offre toutes les fonctions standards des meilleurs Basics actuels avec, en plus, une extrême facilité d'utilisation et une très grande précision de calcul. Un grand magazine informatique, dans un test portant sur les meilleurs matériels actuels, a désigné l'EXL 100 comme le plus précis, grâce à un mode de calcul limitant les erreurs d'arrondi et portant sur 12 chiffres après la virgule (dont 10 affichés). **500 F TTC**

Logiciels
Catalogue sur demande. Plus de 60 nouveaux titres de **69 F à 190 F TTC.**

BON DE COMMANDE

A compléter ou à recopier et à retourner à:
Sté DOM - 274, rue de Créqui 69007 LYON Tél. (7) 872.49.52

Nom, Prénom. _____

Adresse _____

_____

Ville _____ Code Postal ☐☐☐☐☐

Vous passe commande d'un EXL 100, configuration de base au prix de: **3.250 F TTC.** Mode de règlement:

☐ Chèque à la commande
☐ Contre remboursement (coût supplémentaire 53 F)

Signature

LI 12

LIST

LE JOURNAL DES AMATEURS DE PROGRAMMATION

1 COUVERTURE
Qui aurait soupçonné, il y a seulement dix ans, les immenses perspectives auxquelles donneraient accès ces objets de plastique tout bêtes que sont les disquettes ? L'illustration de notre dernière couverture est signée Dominique Le Nouaille.

11 LA GAZETTE DE LIST

19 UNE BONNE RÉSOLUTION POUR LES TRIANGLES DE TOUT ACABIT
Quand aucune courbe n'intervient, tout problème de géométrie peut se réduire à l'étude des triangles qui composent la figure de départ (programme pour ZX Spectrum).

22 LA RÉCRÉ DE LIST
Exercez votre logique et votre ingéniosité pour résoudre quelques problèmes simples en apparence. Qui sait si vous ne parviendrez pas à une solution meilleure que toutes les autres ?

25 RÉPONSES AUX CASSE-TÊTE DU NUMÉRO PRÉCÉDENT

26 COUP D'ŒIL DE LIST
Avec Éditeur de dessins, distribué par IDC, le Dai — réputé depuis longtemps pour ses capacités graphiques — passe à la vitesse supérieure.

28 L'ASSEMBLEUR DU TO 7 ET DU TO 7/70
Pas de programmation en Assembleur si l'on ignore les différents modes d'adressage du microprocesseur. Exemple à l'appui, le comportement du 6809.

30 PASCAL, SUIVEZ LA PROCÉDURE
Il ne faut pas moins de quatre modes de saisie pour lire, en Pascal, les symboles ASCII.

33 PASSEZ-MOI L'EXPRESSION
Comment le Basic fait ses choux gras des parenthèses et des quatre opérations. Une histoire de piles et de priorités.

37 A LA LIMITE, TOUT CONVERGE
Certaines suites mathématiques ne s'approchent de leur limite qu'après longtemps. La méthode d'Euler fait alors gagner beaucoup de temps.

40 SOUS LES COULEURS DE L'AMSTRAD
Le manuel de l'Amstrad ne dit presque rien sur le mode « encre » de la machine. Il est pourtant simple à utiliser et permet des effets variés.

42 HISTOIRE DE LA PROGRAMMATION
Un jour de juin 1833, l'inventeur des machines à différences, Charles Babbage rencontra la fille du poète Byron, Ada de Lovelace. C'est de cette époque sans doute que datent les premiers programmes de l'histoire de l'humanité.

46 LA ROUTINE MÈNE A L'ERREUR
Quand le Canon X-07 signale une erreur de syntaxe, on aimerait bien qu'il liste aussitôt la ligne fautive. Aidez-vous de quelques instructions en langage-machine pour déboguer vos programmes Basic.

48 VOTRE BASIC ET LA BOSSE DES MATHS
Toutes les machines n'ont pas les mêmes capacités mathématiques. Intervalles de calculs, précision, fonctions, bien des choses distinguent les vingt ordinateurs que nous avons testés.

51 DES LISTES COMME VOUS LES AIMEZ
Avec Vic, Pet ou C. 64, en désappointant les pointeurs, on obtient des mises en page sur mesure pour toutes les listes de programmes.

56 EN ESCAMOTANT LES TEMPS DE CALCUL
Très court, rapide comme l'éclair (il y a un truc), un programme en Basic démasque les inconnues des systèmes d'équations linéaires.

58 LOGO-LOTO
Simuler les tirages du Loto sur un ordinateur, c'est enfantin. Tricher, grâce à une bonne connaissance du Logo, c'est médiocre. Mais est-il possible de rendre invisible la supercherie ?

SOMMAIRE

60 **LES PSEUDO-PREMIERS DITS FORTS**
Le « petit théorème » de Fermat permet une approche originale des nombres premiers. Voici un moyen rapide de traiter les nombres jusqu'à dix milliards.

62 **LA BOÎTE A MALICES**
Prenez un programme et retirez-en toutes les astuces, des plus grossières aux plus subtiles. Que reste-t-il ? Rien. Dans ce numéro, des ficelles pour TRS-80, X-07, C. 64, MO 5, TO 7 et 7/70, Dai, Apple II et Sinclair QL.

72 **UNE BASE DE DONNÉES, ÇA CRÉE DES LIENS**
Leurs applications sont innombrables. Apprenons à programmer les bases de données en décortiquant une liste écrite en Basic PX-8.

76 **LE CANON, LA POMME ET LE GUÉPARD**
Les ordinateurs ne s'appellent pas n'importe comment. Remontons aux origines étymologiques de marques et de modèles célèbres.

78 **LE MOT DE LA FIN**
Solutions des casse-tête de la page 22.

Index des annonceurs p. 11.

ÉDITORIAL

A NOS LECTEURS

Vous avez entre les mains le dernier numéro de **LIST** : les journaux, eux aussi, disparaissent. A la plupart, il n'est accordé qu'une existence brève : **LIST** aura vécu un peu plus d'un an. Inclinons-nous.

Nous n'avons pas pu obtenir ce délicat dosage de recettes de vente aux lecteurs et de rentrées publicitaires qui est nécessaire à toute revue, et l'avenir ne semble guère susceptible d'améliorer notablement cette situation.

En effet, le marché de la micro-informatique traverse actuellement une crise dont on ne sait pas trop qui sont en fin de compte les victimes. De déception en mauvaise nouvelle, certains en arrivent même à se demander s'il existe un avenir sérieux pour les machines autres que professionnelles.

Ici, à **LIST**, nous avons toujours considéré l'informatique comme un loisir studieux, une occupation gratuite et de la plus grande utilité. L'arrêt de notre journal ne modifie pas notre point de vue. Il y aura, heureusement, toujours des gens pour aimer et apprendre l'analyse et la programmation. Au bout du compte, c'est sans doute là que l'informatique donne ce qu'elle a de meilleur : elle donne à comprendre et à découvrir.

Les ordinateurs, comme tous les outils, ne valent que par l'usage qui en est fait. Ces outils-là ont une puissance terrible. Il faut que toutes celles et tous ceux qui le souhaitent puissent acquérir une véritable culture informatique, afin que cette culture ne soit pas le monopole de quelques rares spécialistes. Tôt ou tard, on s'en félicitera.

Nous vous remercions de la confiance que vous nous avez toujours manifestée. Nos abonnés verront se poursuivre leur abonnement sur la revue **L'Ordinateur Individuel.**

LA RÉDACTION

RÉDACTION-RÉALISATION
Directeur de la rédaction : Bernard Savonet
Rédacteur en chef : Jean Baptiste Comiti
Responsable de rubrique : Anne-Sophie Dreyfus
Rédacteur : Maryse Gros
Conception graphique et secrétariat de rédaction : Eliane Gueylard
Administration : Marie-Hélène Muniz

Ont collaboré à ce numéro : Michel Aubry, Pierre Barnouin, Pascal Baube, François J. Bayard, Frédéric Blanc, Pierre Brandeis, Michel Brochand, Laura Campagnet, Jean-Paul Carré, Thierry Chamoret, Christophe et Etienne Chantraine, Vincent-Pierre Comiti, Raymonde Coudert, Robert Daguesse, Augustin Garcia, Francis Goudard, Laurent Gras, Pierre Ladislas Gedo, Renée Koch, Bernard Kokanosky, Jean-Pierre Lalevée, Hervé Le Marchand, Franck-Olivier Lelaidier, Sylvain Lemaire, Alain Mariatte, Lucien Strebler, Henri Volleau, André Warusfel.

Illustrations : Philippe Burel, Antoine Chereau, Dik, Frapar, Bernard Helme, Dominique Le Nouaille, Alain Mangin, Alain Prigent, Nicolas Spinga.

ÉDITION-PUBLICITÉ-PROMOTION
Éditeur : Jean-Paul Nizard
Éditeur-adjoint : Jean-Daniel Belfond
Administration : Maryse Marti, assistée d'Anne Stolkowski
Publicité : Bénédicte Lizon, assistée de Marie-Christine Jugeau

VENTES
Diffusion NMPP : Béatrice Ginoux Defermon
Abonnements : Muriel Watremez assistée de Cécilia Mollicone

Directeur de la publication : Jean-Luc Verhoye

LIST est une publication du groupe tests
5 place du Colonel Fabien - 75491 Paris Cédex 10
Téléphone : (1) 240 22 01 - Télex : LORDI 215 105 F

Tarif d'abonnement (10 numéros)
France : 160 FF TVA 4 % incluse, Belgique : 1330 FB, Suisse : 50 FS, autres pays : 210 FF (surface), 261 FF (avion).

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'Art. 41, d'une part que « les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective », et, d'autre part, que les analyses et les courtes citations dans un but d'exemples et d'illustrations, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants-droit ou ayant-cause est illicite » (alinéa 1er de l'Art. 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contre-façon sanctionnée par les Art. 425 et suivants du Code Pénal.

BVP Bureau de Vérification de la Publicité.
Notre publication contrôle les publicités commerciales avant insertion pour qu'elles soient parfaitement loyales. Elle suit les recommandations du Bureau de Vérification de la Publicité. Si, malgré ces précautions, vous aviez une remarque à faire, vous nous rendriez service en écrivant au BVP, BP 4508, 75362 PARIS CEDEX 08.

LA GAZETTE DE LIST

CHEZ LE LIBRAIRE

Guide du graphisme MSX
Mike Shaw
Editions Sybex
1985, 184 pages
Prix : 98 FF

L'ÉTUDE tourne autour du « VDP », ou processeur d'affichage vidéo, par lequel on accède à toutes les fonctions graphiques du MSX. La formation des caractères, la couleur, les lutins et les quatre modes-écran. Le propos est illustré par des exemples de programmes, en Basic et en langage-machine.

Amstrad, graphismes et sons du CPC 464
Jorg Walkowiak
Editions Micro Application
1985, 184 pages
Prix : 129 FF

L'AMSTRAD est riche en possibilités graphiques et effets sonores, dont la mise en œuvre n'est pas toujours simple. Ce livre en démystifie les principaux aspects. Le lecteur y trouvera aussi d'autres applications concernant les jeux et les utilitaires (DAO, calcul de coordonnées, dessin en deux ou trois dimensions, synthétiseur, etc.).

Exercices en Basic pour Yeno-Sega 3000
Frédéric Levy
Editions du PSI
1985, 154 pages
Prix : 95 FF

Exercices en Basic pour Alice et Alice 90
Maurice Charbit
Editions du PSI
1985, 146 pages
Prix : 95 FF

DANS les deux cas, il s'agit de réaliser des programmes en partant d'éléments précis (énoncé d'un problème, données à entrer au clavier, résultats à obtenir). Les exercices sont variés : manipulation de tableaux, traitement de chaînes de caractères, histogrammes, jeux... Une solution est fournie à la suite de chaque problème.

Micro-V.O. la Micro passion

LES deux revues **Micro 7** et **Votre Ordinateur** se rapprochent pour n'en faire plus qu'une : **Micro-V.O.** Résolument orienté vers le grand public, notre confrère a choisi un sous-titre coup de cœur : le magazine de la micro passion. Leur premier numéro est dans les kiosques le 19 septembre au prix de 20 FF. ■

Can'ell ou comment un Minitel devient le moniteur du Canon X-07

POUR devenir un véritable ordinateur de table, il ne manque au Canon X-07 qu'un véritable écran vidéo. Celui du Minitel, par exemple, ne coûte pas cher. Il est facile à trouver (dans les agences commerciales des PTT) et il dispose de 19 lignes de 40 caractères. La liaison est désormais possible grâce à *Can'ell*, un ensemble comprenant un cordon de raccordement qui va du Canon X-07 au Minitel, et un logiciel sur cassette.

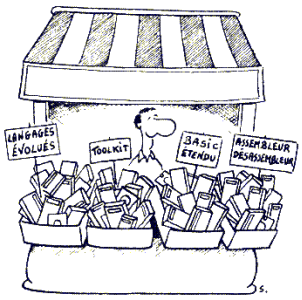
Le logiciel implante un programme en langage-machine qui définit le nouvel écran du Canon sur le Minitel. Et ce dernier ne consomme alors rien d'autre que de l'électricité, à moins d'être utilisé comme un modem.

Sur la même cassette, deux programmes de jeux font la démonstration de l'efficacité de Can'ell. Le premier est une forme simplifiée du « Compte est bon » et le second consiste à diriger une chenille, à partir du pavé de flèches du X-07, afin qu'elle ramasse des champignons dans un labyrinthe.

Can'ell a été conçu par la société **Feeling Soft** qui est actuellement la seule à distribuer ce produit. L'ensemble câble et logiciel coûte 390 FF ttc (250 FF pour le câble et 180 FF pour le logiciel).

Feeling Soft
701 Boulevard Haute Folie
14200 Hérouville St-Clair
et 4 rue de la Cardonnière
14000 Caen ■

LOGICIELS



M.A. Base
Gestion de fichiers
Cassette pour Amstrad
Edité par Micro Application
Prix : 165 FF

LE logiciel **M.A. Base** est livré avec un carnet d'adresses prêt à l'emploi, mais on peut bien entendu créer ses propres fichiers en définissant son masque de saisie (place des rubriques sur la fiche). On dispose des principales options rencontrées habituellement sur ce type de programme : recherche, modification, tri et bien sûr impression. Les fichiers sont sauvegardés sur cassette.

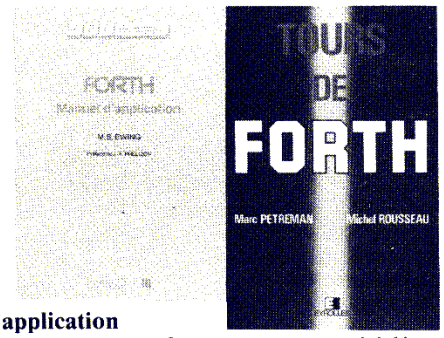
Pascal
Microcassette
pour Sinclair QL
Edité par Computer One
Distribué par Sinclair/Direco
Prix : 850 FF

CE Pascal est au standard ISO à quatre exceptions près, signalées dans le manuel. Il utilise toutes les possibilités propres au QL. Une option « Make-job » compile tout programme Pascal en un fichier directement exécutable sous QDOS, ce qui permet un fonctionnement en multitâche. Ce logiciel est incontestablement agréable à utiliser car on est toujours sous un menu.

INDEX DES ANNONCEURS

Casio ..... p. 84
Dynamit Computer ..... p. 15
Exelvision ..... p. 2,3
KBI ..... p. 77
L'Ordinateur Individuel ..... p. 82, 83
Maubert Electronic ..... p. 13
Microstrad ..... p. 81
Microtom ..... p. 75
Micro V.O. ..... p. 6, 7, 8, 9
POM'S ..... p. 80
PSI ..... p. 10

DEUX LIVRES



Forth, manuel d'application
M.S. Ewing
Editions Masson
Collection ABC des langages
1984, 120 pages
Prix : 85 FF

LA collection ABC des langages est destinée à des lecteurs qui souhaitent « approfondir leurs connaissances », mais qui n'ont pas « d'expérience réelle de la programmation ». Or, ce manuel d'application est peut-être un peu abrupt pour eux. L'apprentissage du Forth nécessite un grand nombre de schémas représentant les piles, l'organisation de la mémoire, etc., et il y en a assez peu.

Après une introduction sur l'historique et les caractéristiques du langage, ce sont les notions de mots, de piles et de gestion de mémoire de masse qui sont étudiées (il sera utile de se reporter simultanément au vocabulaire du standard Forth 79 présenté au chapitre 4). Les mécanismes internes du langage sont ensuite abordés de manière détaillée et c'est ici que le lecteur aura besoin de connaître le Forth. Le dernier chapitre, enfin, traite de la mise en œuvre du langage sur de gros systèmes.

Ces pages seront de préférence mises entre les mains d'un lecteur averti, le débutant risquant peut-être de se décourager.

Tours de Forth
Marc Petreman
Michel Rousseau
Editions Eyrolles
1985, 174 pages
Prix : 98 FF

IL ne s'agit plus ici d'apprendre le Forth. Aucune étude en effet sur les primitives de base mais une série de programmes, directement exploitables et bien expliqués, précédée d'un chapitre traitant des algorithmes et des organigrammes.

Ces programmes ont été développés sur un Oric et leur adaptation sur une machine sans mode texte telle que le Hector HRX (qui ne dispose que d'un mode d'affichage haute résolution) sera certainement délicate. Néanmoins, le lecteur en tirera profit, quelle que soit sa machine car les astuces de programmation seront utilisables pour d'autres applications. Parmi celles qui se trouvent dans le livre, notons une fonction catalogue (VLIST amélioré), un vidage mémoire, un décompilateur et des programmes graphiques (histogrammes, jeu de taquin, créations de lutins) qui permettront de comprendre comment l'écran est géré. Et pourquoi ne pas essayer de réaliser votre premier jeu d'aventure en Forth à partir des explications fournies par les auteurs ?

La fin de l'ouvrage est consacré au nouveau standard Forth 83. On y trouvera toutes les différences entre Figh Forth, Forth 79 et Forth 93 et la manière d'obtenir les nouvelles définitions à partir des anciennes. Un livre concret qui apportera au programmeur initié une foule d'idées à mettre en pratique.

MB ■

EN LIBRAIRIE

Guide pratique de l'Apple IIc
Bruno de Latour
Editions Cedic/Nathan
1985, 140 pages
Prix : 89 FF

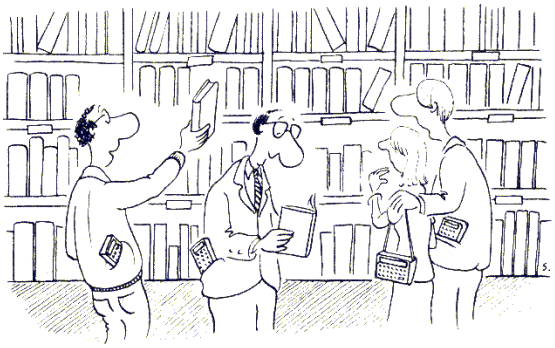
POUR se familiariser avec l'Apple IIc et son environnement : présentation de l'appareil, du DOS 3.3 et de Prodos. On y explique comment utiliser les disquettes système. Des photos d'écran illustrent les commentaires. Un aperçu des différents logiciels et périphériques disponibles dans le commerce est proposé.

Les périphériques de l'Atari
Daniel-Jean David
Editions du PSI
1985, 130 pages
Prix : 85 FF

L'OUVRAGE passe en revue l'utilisation des périphériques des modèles 400, 800, 600 XL, 800 XL : lecteur de cassettes, imprimantes, lecteur de disquettes, manettes de jeux, tablette à digitaliser et crayon optique. L'auteur insiste plus particulièrement sur leurs applications (gestion de fichiers ou de bases de données, etc).

Commodore 64, Basic approfondi
Gary Lippman
Editions Sybex
1985, 208 pages
Prix : 128 FF

IL s'agit de la suite de « Commodore 64, premiers programmes » de Rodnay Zaks. Le lecteur a déjà passé en revue les principaux éléments de base et il fait ici connaissance avec des notions plus élaborées telles que l'utilisation des fichiers sur disque et sur cassette, les tris, les recherches, la programmation modulaire, etc.



Standard MSX deuxième génération

MAI 1985, Tokyo : les sociétés **Microsoft** et **ASCII** annoncent un nouveau standard MSX. Entièrement compatible avec les ordinateurs de la première génération, la seconde version conserve le même microprocesseur, un Z 80A. Sa mémoire morte augmente et passe à 48 Ko tandis que le minimum imposé pour la mémoire vive de base sera cette fois-ci de 64 Ko.

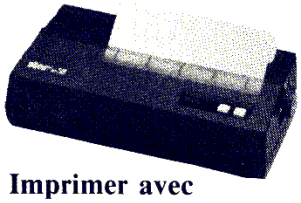
Le Basic est encore plus « étendu » bien sûr, mais l'originalité de MSX 2 réside sans conteste dans le processeur vidéo pour lequel la mémoire d'écran, 64 Ko, peut être portée à 128 Ko, ce qui est considérable (résolution graphique maximum : 512×212 points pour seize couleurs).

Toujours d'origine, une horloge interne permanente et un crayon optique, alors que les concepteurs ont prévu, en option cette fois, l'adjonction d'un circuit intégré dotant la machine de qualités sonores remarquables et la possibilité de digitaliser une image vidéo.

Reste à savoir quels seront les constructeurs qui joueront cette carte et à quel moment le nouveau standard sera introduit sur le marché européen. Quelques noms ont déjà été cités **(Toshiba, Yamaha)** ; et l'équivalent en francs français des prix annoncés au Japon place le nouveau matériel dans une fourchette allant de 4 000 à 5 000 francs. Toutes ces informations viennent encore de bien loin, elles devraient être bientôt complétées par d'autres, plus précises. ■

Sir Clive Sinclair passe la main

FIN mai, la société **Sinclair** recherchait des partenaires. Elle avait besoin d'un apport de capitaux nouveaux (10 à 15 millions de livres) pour financer son expansion à long terme et ses projets de restructuration. Elle les a trouvés : Hollis, filiale du groupe Pergamon Press (Daily Mirror), qui a versé 12 millions à la firme britannique détient maintenant 75 % du capital. Sir Clive, remplacé dans sa fonction par Bill Jeffrey, précédemment Directeur de la division TV de Sinclair, reste Président honoraire à vie. Il se consacrera désormais à la recherche. ■



Imprimer avec MSX

L'INTERFACE parallèle de l'imprimante ST-80 autorise sa connexion avec les ordinateurs MSX. Elle est thermique et travaille sur 80 colonnes. Sa vitesse d'impression est de 60 caractères par seconde. La Star ST-80 vaut 2 500 FF ttc et elle est distribuée par **Hengstler.**

Hengstler
94-106 rue Blaise Pascal - BP 71
93602 Aulnay sous Bois
Tél. : (1) 866 22 90 ■

Duriez se télématise

EN composant le (1) 329 42 99 (à partir du 25 octobre 1985, ce sera le 43 29 42 99) avec un Minitel, vous entrez en contact avec la boutique **Duriez.** Pour chacun des articles disponibles — ordinateurs de poche, familiaux, périphériques, logiciels —, on obtient les tarifs qui sont réactualisés tous les huit jours. Attention toutefois, ces renseignements ne sont fournis qu'à titre indicatif...

Le coût de la consultation est équivalent à celui d'une communication téléphonique classique. ■

Philips au standard MSX

LES derniers-nés de **Philips,** les VG 8010 et VG 8020, tous deux au standard MSX, sont apparus sur le marché dans le courant du mois de juillet respectivement à 2 290 et 2 990 F environ (ces prix ne comprennent que l'unité centrale). Outre les 16 Ko pour la gestion de l'écran, le premier modèle offre 32 Ko de mémoire vive ; le second, un peu plus musclé, dispose de 64 Ko (+ 16 Ko de mémoire écran) et d'un clavier mécanique.

Deux lecteurs de disquettes devraient être disponibles en octobre (environ 1 890 et 3 190 FF). Enfin, un Logo et un Pascal UCSD sont annoncés ; ils seraient actuellement en cours de développement. ■

CHEZ LE LIBRAIRE

Programmatique
Chantal et Patrice Richard
Editions Belin
Collection Modulo
1985, 192 pages
Prix : 90 FF

LA « programmatique », néologisme créé par les auteurs, désigne la méthodologie de la programmation, indépendamment de tout matériel ou langage particulier, à l'exception toutetois d'ALADIN, langage inventé également par Chantal et Patrice Richard (ALADIN, lisez ALgorithmique Arborescente pour Débutants en INformatique). Un ouvrage spécialement destiné aux élèves, étudiants ou enseignants.

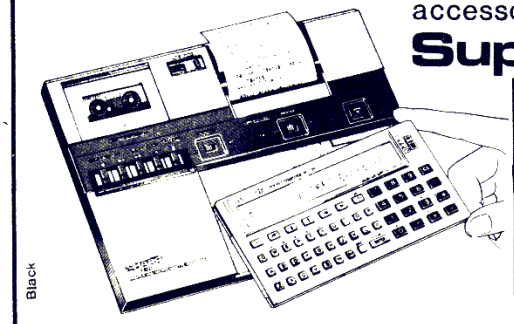
Electro-Basic
Programmes de simulation en Basic
Claude Nowakowski
Editions du PSI
1985, 158 pages
Prix : 95 FF

NI initiation à l'informatique, ni cours d'électronique, cet ouvrage a pour but de fournir aux étudiants en électronique des applications qui permettent d'utiliser l'ordinateur. Il s'adresse à ceux qui sont intéressés par la théorie des circuits. Après un chapitre consacré à des généralités sur les réseaux R,L,C, l'étude porte sur les fonctions périodiques.

CALCULATRICES et ORDINATEURS de POCHE

accessoires et machines à écrire électroniques Sharp-Canon

Super Promotion Rentrée!



SHARP	HEWLETT-PACKARD
PC 1246-1247-1260-1261 PC 1401- 1402-1450-1350-1500A	Remise supplémentaire –20% sur HP 11-12-15
CANON	**CASIO**
X 07 et périphériques etc. V20	FX700-FX702P-FX750-PB200 PB700- etc.

NOUVEAUTE "M.S.X "EN DEMONSTRATION

MAUBERT ELECTRONIC 49, bd. St Germain. PARIS 5° TEL. 325.88.80 PLACE ET M° MAUBERT

LA GAZETTE DE LIST

CHEZ LE LIBRAIRE

Le langage Pascal Iso
Thierry Chamoret
Editions Editests
1985, 224 pages
Prix : 130 FF

L'AUTEUR a réalisé un instrument de travail très complet, exposant les particularités et possibilités du langage, à la fois manuel de référence et guide décrivant le fonctionnement interne des instructions.

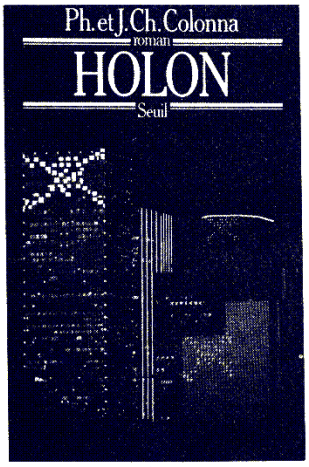
Il s'adresse à des lecteurs ayant déjà de bonnes connaissances en informatique et maîtrisant au moins un langage de programmation. Quant à ceux qui connaissent Pascal, qui ont cessé de le pratiquer pendant un moment et qui désirent s'y remettre sérieusement, ils auront avec cet ouvrage une bonne occasion de se rafraîchir la mémoire.

Programmes pour MSX
Serge Pouts-Lajus
et Pierre Champeaux
Editions Cedic/Nathan
1985, 122 pages
Prix : 75 FF

LES programmes rassemblés utilisent surtout les possibilités graphiques et sonores des MSX. On y trouve des morceaux de musique comme « le derviche tourneur » de Marcel Dadi. Il y a aussi quelques utilitaires et, en annexe, un lexique du Basic MSX.

UN LIVRE

Holon
roman
Philippe et Jean-Christophe Colonna
Editions du Seuil
1985, 324 pages
Prix : 89 FF



NOUS n'avons jamais signalé d'ouvrages de fiction dans ces colonnes. Nous ferons donc une petite exception pour **Holon**, roman qui nous transporte à la fin du XXIe siècle. La grande révolution informatique (dont on parle tant et tant de nos jours) a alors eu lieu. Les ordinateurs et la télématique ont depuis longtemps infiltré, investi, contaminé la vie quotidienne. Présence familière, plutôt bénéfique à première vue, mais implacable. Le fichage est ainsi devenu l'un des éléments les plus importants de la vie du citoyen. Sur chacun pèse en effet la menace d'être radié des fichiers, c'est-à-dire d'être rejeté, clochardisé dans un monde marginal et cauchemardesque.

On peut avoir de l'informatique une vision optimiste. Ce n'est pas celle des deux auteurs d'**Holon** et leur hypothèse tient assez bien debout. Si vous voulez vous faire peur en plongeant dans l'enfer informatisé, et si vous aimez les romans bien ficelés, je vous conseille celui-là. Il démarre un peu lentement à mon goût, mais une fois que la mayonnaise prend, quelle histoire !

JBC ■

Amstrad encore plus gros

LE nouveau modèle d'ordinateur d'**Amstrad** porte le nom de : CPC 6128. Il est riche de 128 Ko de mémoire vive. Comme le CPC 664 — le précédent modèle — il dispose d'un lecteur de disquettes de 3 pouces (disquettes d'une capacité de 340 Ko, formatées). Son clavier est moins encombrant que celui de son prédécesseur, le pavé numérique étant accolé aux autres touches. Les logiciels sur cassettes ou sur disquettes, conçus pour les précédents modèles, tournent sur celui-ci. Il est vendu au prix de 4 490 FF ttc avec un moniteur noir et blanc, et 5 990 FF ttc avec un moniteur couleur.

Les prix des autres modèles ont diminué : le CPC 464, 64 Ko de mémoire vive et un lecteur de cassettes intégré, coûte 2 690 FF ttc avec un moniteur noir et blanc, et 3 990 FF ttc avec un moniteur couleur ; le CPC 664, 64 Ko de mémoire vive et un lecteur de disquettes intégré, coûte 3 790 FF ttc avec le moniteur noir et blanc, et 5 290 FF ttc avec le moniteur couleur. ■

Carte moniteur XP-140 pour Canon X-07

LA XP-140, comme toutes les cartes d'application du Canon X-07, s'insère au dos de la machine. A ses 8 Ko de mémoire morte s'ajoutent 4 Ko de mémoire vive sauvegardés par pile, dont 1 Ko utilisé par le moniteur.

Cette carte a deux fonctions : elle étend le Basic du X-07 et offre un mode moniteur (d'où

Le Tandy 200 succède au Modèle 100

AVEC un écran de seize lignes de 40 caractères, le portable **Tandy 200** reste compatible avec son prédécesseur, le Modèle 100. Sa mémoire vive de 24 Ko de base (dont 19 seulement pour l'utilisateur) peut être étendue jusqu'à 72 Ko par l'adjonction de modules de 24 Ko. Comme le modèle 100, le Tandy 200 a ses logiciels intégrés : un tableur (Multiplan), un traitement de texte (Text), un programme de gestion d'adresses. Il est vendu 8 250 FF ttc, chaque module de mémoire vive valant 1 720 FF ttc.

Le modèle 100, toujours en vente, coûte quant à lui 4 495 FF ttc. ■

son nom) pour créer des programmes en langage-machine.

A la mise en route, on dispose sous Basic de neuf nouvelles instructions pour la gestion de fichiers (BSAVE et BLOAD pour les fichiers binaires, BLOADI, MERGE et RENAME), l'exécution de routines en liaison avec le moniteur (MEXEC, USRM) et l'édition de programmes en Basic (RENUM, DEL).

Le mode moniteur est à l'évidence l'atout de cette carte et justifie aisément son acquisition : une commande, MON, lui passe la main, une autre BAS, la rend au Basic. Il permet de visualiser ou d'écrire des données en mémoire (M,V), de traiter les registres Z 80 (R), de désassembler (VN), d'exécuter (G), de suivre des routines en langage-machine (T,N). Les vidages et les désassemblages ne sont pas limités à l'écran, l'instruction PM permettant leur édition, en hexadécimal ou en caractères ASCII, sur l'imprimante. Certaines commandes, plus spéciales, rendent possibles la pose de points d'arrêt (BP), et leur suppression (K). Enfin, les entrées/sorties se font sur un octet avec IN et OUT. Au total, quinze commandes dont une pour le retour au Basic. Notons que le petit écran du X-07 est particulièrement bien utilisé, par exemple lors de l'affichage de l'ensemble des registres (PC, SP, AF, BC, DE, HL, IX, IY).

Deux manuels sont fournis avec la carte. Le premier est un guide de l'utilisateur qui décrit, avec exemples, les nouvelles instructions Basic et les commandes du moniteur. Le deuxième est un « guide technique » qui, s'il n'est pas très facile d'accès, recèle de précieuses informations pour la programmation. On y trouvera les principales routines système, les commandes du deuxième processeur, la gestion des interruptions, etc.

Malgré l'absence regrettée d'un assembleur, la carte XP-140 (avec sa documentation) est un bon outil, particulièrement utile pour la mise au point de programmes en langage-machine.

LG ■

LOGICIELS

Atari Logo
Cartouche pour Atari 130 XE
Edité par Atari
Prix : 800 FF

DEUX manuels sont livrés avec la cartouche : le premier est une introduction à la programmation en Logo, le second est un ouvrage de référence qui s'adresse aux utilisateurs connaissant déjà le langage. Un petit fascicule, qui récapitule les primitives du Logo Atari, vient compléter cette documentation.

Macro-assembleur
Disquette pour Atari 130 XE
Edité par Atari
Prix : 450 FF

LA disquette contient un macro-assembleur conditionnel et un éditeur. L'utilisateur dispose ainsi d'un moyen de se créer une bibliothèque de macro-instructions qui seront assemblées à sa demande. Ces dernières peuvent s'emboîter et offrent donc les avantages d'un langage de programmation de haut niveau. Ce produit s'adresse plutôt aux développeurs et aux amateurs avertis.

Logo V1.0
Cassette pour Oric-1/Atmos
Edité par Loriciels
Prix : 295 FF

UNE version du langage créé par Seymour Papert, adaptée à l'Oric. On y retrouve les principales caractéristiques : tortue, mots-clés, procédures, récursivité. Comme pour tous les logiciels de ce type, la notice n'a pas pour vocation l'initiation au Logo. Une formation préalable est nécessaire.

Diskettes promo 5.25" SF/DD/10 pc.	75 F
Diskettes NASHUA 5.25" SF/SD/10 pc.	100 F
Diskettes 3M 5.25" SF/DD/10 pc.	175 F
Diskettes couleur + boîte plastique/10 pc.	170 F
Diskettes 3" 1/2 par 10	270 F
Diskettes haute densité	Téléphonez.

ORDINATEUR COMPATIBLE IBM PC-XT !
— 2 Drives 360 K-256 K résident
— 1 carte Monoch. graph. imprimante
— 1 carte drive
— Clavier AZERTY
Le tout (TTC) **9 900 F**

Pour APPLE

Carte 128 K	900 F
Carte 16 K langage	350 F
Carte 80 col. (2+)	540 F
Carte 80 col. + 64 K (2e)	500 F
Softswitch 40/80 col.	150 F
Wildcard déplombage (2+)	350 F
Speech card	290 F
Z-80 CP/M	300 F
Mocking card	750 F
Musique Synthé. (9 voix)	550 F
5069 Via	450 F
Drive Slim 2+, 2e	1 250 F
Connecteur Drive 2c	80 F
Joystick	165 F
Tablette graphique APPLE/IBM	950 F
Ordinateur CPU 64 K (sans ROMS) compatible APPLE 2e	3 900 F
Kit de ROMS + 65CO2 + cabochons Français (Original APPLE)	1 100 F

Autres

Modem CCITT V 21 (Universel)	790 F
Tablette graphique COMMODORE	650 F
Imprimante MT-80 + (100 CPS)	2 750 F

N'ATTENDEZ PAS. INVESTISSEZ DANS VOTRE FUTUR !

Moniteur monochrome IBM	1 450 F
Disque dur 10 Mega avec controleur	8 895 F
Imprimante MT-80 PC (130 cps)	3 990 F
Imprimante SG-10 (120cps)	3 850 F

JACKINTOSH !!!
— Unité centrale 512 K
— Lecteur disquettes 500 K
— Moniteur Monochr. haute résolution
— et en plus la souris !!!
— Cadeau de 10 diskettes 3"1/2 aux 500 premiers acheteurs
L'ensemble **9 950 F**

DYNAMIT COMPUTER
MOINS CHER QUE MOI TU MEURS !!!
54, rue de Dunkerque - 75009 Paris
Métro : Anvers. Tél. : (1) 282.17.09
Heures d'ouverture : Lundi au samedi de 9 h 30 à 19 h
Écrivez pour une liste plus complète

LA GAZETTE DE LIST

CHEZ LE LIBRAIRE

Clefs pour MSX
Tome 1 - Système de base
Rémy Pineau
Editions du PSI
1985, 270 pages
Prix : 150 FF

CE pavé (reliure spirale) se veut « livre de bord » du MSX. L'ordinateur est disséqué, ses connecteurs, processeurs, instructions Basic, microprocesseur et adresses — mémoires morte et vive — passés en revue. Les nombreux tableaux récapitulatifs à plusieurs entrées et les tables de références de toutes sortes sont complétés d'un index des abréviations anglais-français.

Apprendre à programmer
Jack Emmerichs
Editions Belin
Collection Modulo
1985, 264 pages
Prix : 160 FF

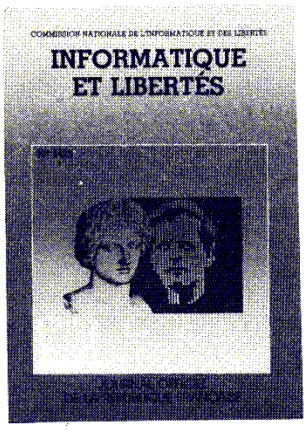
COMMENT concevoir ses programmes avec méthode. Ce n'est ni le langage utilisé, ni le matériel qui préoccupe l'auteur (même si quelques exemples en Basic ou en Pascal apparaissent çà et là), mais les différentes étapes qui mènent à la réalisation d'un programme, de l'idée de départ à la mise en forme définitive.

Mathématiques et graphismes
Micro pour l'école
Gérald Grandpierre
Richard Cotté
Editions du PSI
1985, 260 pages
Prix : 130 FF

LES droites (déformations, rotations, fractals), les courbes (approximations de fonctions, systèmes différentiels) et l'espace (surfaces, manipulations d'objets) : ce sont les trois grands chapitres du livre.

Les programmes proposés sont susceptibles d'être adaptés à différents modèles d'ordinateurs, ceux-ci devant bien évidemment disposer de possibilités graphiques.

L'informatique et les libertés



LA Commission Nationale de l'Informatique et des Libertés (CNIL) publie une remise à jour des textes législatifs relatifs à « la protection des personnes à l'égard du traitement automatique des données » (7e édition - 30 avril 1985). Il s'agit d'un recueil de 235 pages que l'on peut se procurer pour 45 FF (+ 7 FF de port) auprès de la Direction des Journaux Officiels. La brochure s'intitule « Informatique et Libertés » et porte le n° 1473.

Direction des Journaux Officiels
26 rue Desaix
75727 Paris Cedex 15
Tél. : (1) 578 61 39 poste 715
Minitel : 16 (3) 615 91 77 + JOEL 5 ■

Un club pour l'Amstrad

SI l'adhésion à l'Association pour la Promotion du CPC est gratuite, elle ne donne cependant pas droit à sa lettre bimensuelle. Pour recevoir cette dernière, il faut s'abonner : 200 FF pour vingt numéros par an (le premier numéro est envoyé contre 10 FF). L'APC prévoit, entre autres, la réalisation d'un annuaire regroupant les utilisateurs de l'Amstrad et un guide de logiciels.

On peut leur écrire à l'adresse suivante, en n'oubliant pas une enveloppe timbrée et libellée :
Association pour la Promotion du CPC
109 rue Gaston Lauriau
93100 Montreuil
Tél. : (1) 859 71 01 ■

Vacances d'hiver et micros

L'ARDÈCHE offre-t-elle des températures clémentes pendant les mois d'hiver ? Le Microtel club de la région propose de venir sur place le découvrir dans le cadre de stages d'informatique ouverts à diverses activités de loisirs (randonnées pédestres, photographie, etc.).

Trois sessions sont prévues : du 27 novembre au 3 décembre 1985, coût 1 800 F ; du 22 au 29 décembre et du 26 décembre au 1er janvier coût 2 800 F la session, réveillon compris.

Tous ces tarifs (des réductions peuvent être accordées aux

UNE CASSETTE

LM Plus
Compilateur Basic
Cassette pour Oric-1 et Atmos
Edité par Isosoft
Prix : 250 FF

LE Basic, langage interprété, ne brille pas spécialement par sa rapidité ; ses avantages résident plutôt dans sa souplesse d'emploi et la diversité de ses instructions. Le compilateur transforme le Basic en langage-machine et lui confère du même coup la rapidité de ce dernier.

Après avoir chargé *LM Plus*, on entre une liste Basic en mémoire, on appelle l'utilitaire par un CALL # 9200 et le tour est joué. Le processus de compilation se décompose en cinq étapes : le programme regroupe les DATA et construit une table d'adressage pour les numéros de lignes référencés, il dimensionne ensuite les tableaux (DIM), il affecte à chaque variable une place en mémoire et il compile ; enfin, il achève de coder les numéros de lignes. L'utilisateur suit, s'il le veut, ces différentes opérations (qui durent quelques secondes) car l'écran indique le numéro de l'étape en cours.

LM Plus propose ensuite d'afficher la table des adresses des variables et donne celles de début et de fin du code compilé. Celui-ci se lance avec un CALL suivi de l'adresse de départ. Ainsi transformé, il tourne environ sept fois plus vite que son « original » en Basic.

Pouvons-nous pour autant compiler tous nos programmes ? Hélas non ! Pour tenir dans la mémoire de l'Oric sans prendre toute la place, on a sévèrement limité *LM Plus*. La première restriction est de taille : on ne peut travailler qu'avec des entiers (de −32768 à +32767). Voilà qui écarte d'emblée tous les calculs mathématiques et limite sérieusement les autres. La seconde restriction concerne les expressions. Elles ne peuvent se composer que de deux opérandes au maximum. Si A = B + C est permis, A = B + C + D provoque l'apparition d'un message d'erreur. Il faut décomposer l'expression : A = B + C et A = A + D. Les programmes déjà écrits devront donc être remaniés dans ce sens.

Enfin, *LM Plus* n'accepte qu'un seul IF sur une même ligne et ne semble pas admettre les conditions sur les chaînes de caractères. Les fonctions mathématiques sont bien sûr interdites, de même que DEF FN, USR, STORE, RECALL et quelques autres.

Mais le produit a aussi ses bons points. On peut sauvegarder le programme compilé et ses variables, et il n'est pas nécessaire d'avoir *LM Plus* en mémoire pour le recharger et le faire fonctionner. Il suffit de noter l'adresse d'implantation pour le lancer avec un CALL. Il est également possible de définir soi-même l'adresse d'implantation. Avec la table des variables, on pourra « mélanger » des parties de programme compilé et de Basic. D'autant que la présence du compilateur en mémoire n'empêche absolument pas de lancer l'exécution d'un programme en Basic, d'où une grande facilité de mise au point. Il faut simplement éviter de passer en haute résolution : cela conduirait à effacer l'utiliaire.

LM Plus est accompagné d'une notice claire et assez complète qui donne toutes les syntaxes admises pour chaque instruction. Il sera particulièrement utile à ceux qui veulent faire de l'animation, domaine où le langage évolué (bien que travaillant en général sur des entiers) est vraiment trop lent.

PB ■

familles) englobent l'hébergement dans une maison familiale et l'adhésion à Microtel pour l'année scolaire 1985/86.

Dossier d'inscription et renseignements complémentaires au : (75) 39 18 80.

Microtel Ardèche Sud
La Croix de Malet - BP 36
07110 Largentière ■

Apprendre le Pascal

LA société **Borland** présente, sous la forme d'un manuel et d'une disquette, un cours d'initiation au Pascal. Son nom, *Turbo Tutor*, rappelle celui de son prédécesseur, Turbo Pascal. Le produit est destiné à guider le débutant dans son approche de ce dernier ; mais il propose aussi, à l'intention du programmeur confirmé, des éléments pour une utilisation avancée du langage.

Il coûte environ 480 FF et il est commercialisé par **Fraciel**, qui distribue aussi Toolbox (750 FF), une série de trois utilitaires pour Turbo Pascal (gestion de fichiers, tri, module d'implantation de routines). Tous deux tournent sous les systèmes d'exploitation MS-DOS, PC-DOS et CP/M.

Fraciel
42 rue des Prébendes
37000 Tours
Tél. : (47) 64 08 52 ■

Canon équipe les MSX

LA société **Canon** s'intéresse aux MSX. Elle annonce un lecteur de disquettes, le VF 100 qui devrait être commercialisé courant septembre. Il se connectera sur un port cartouche du MSX et les disquettes (3 pouces 1/2) auront une capacité de 720 Ko après formatage. Livré avec son contrôleur et une disquette MSX-DOS, le VF 100 coûtera environ 3 400 FF ttc.

Autre périphérique MSX, la T22A est une imprimante thermique qui va jusqu'à 140 colonnes. Sa vitesse d'impression sur 80 colonnes est de 56 caractères à la seconde et elle vaut environ 2 000 FF ttc.

Une interface X 740 permet de connecter un Canon X-07 à un MSX pour transférer des données d'un matériel à l'autre. Son prix : environ 450 FF ttc. ■

Microstrad Tout Amstrad

MICROSTRAD (comprenez : un magazine entièrement consacré à la famille Amstrad) sort le 16 septembre. Les adeptes du CPC devraient y trouver une mine de renseignements sur leur matériel. Exploiter les possibilités de la machine, améliorer ses performances, dévoiler ses ressources cachées, découvrir de nouvelles applications, autant de domaines privilégiés pour cette revue d'une soixantaine de pages que votre marchand de journaux vous cèdera pour 28 FF. ■

Les souris de la rentrée seront-elles à l'heure ?

En France
Atari 520 ST

L'ATARI 520 ST arrive en France. Il devrait être en vente depuis le 1er septembre, au prix de 10 000 FF ttc. Dans l'emballage, à côté de l'unité centrale, on trouvera le moniteur noir et blanc, un lecteur de disquettes, une souris, un système d'exploitation GEM (Graphics Environment Manager) sur disquette, et quatre logiciels : Gempaint (logiciel graphique), Gemwrite (traitement de texte), Logo et Basic. Si ce dernier n'était pas encore disponible (il est conçu par Digital Research), une feuille préviendrait l'utilisateur : il pourra se procurer le Basic ultérieurement sans bourse délier. Quant au système d'exploitation GEM, il est encore sur disquette dans une version qui, probablement, n'est pas définitive. Si tout va bien, il devrait être disponible en mémoire morte vers le début novembre. A ce moment-là, il sera possible de le faire installer, mais on ne connaît pas encore les modalités de cette transformation.

Aux Etats-Unis
L'Amiga

PLUSIEURS revues anglo-saxonnes *(Personnal Computer World, Compute !, Bytes,* etc.) ont présenté, pendant l'été, les aspects techniques du nouveau Commodore appelé Amiga : 256 Ko de mémoire vive extensible à 512, un processeur Motorola 68000, un lecteur de disquettes intégré, un moniteur haute résolution et de qualité, devant gérer 256 000 points (640 × 400) dans le mode maximum et 4 096 couleurs. La gestion de plusieurs tâches simultanées est un atout (remarqué par tous) de cette machine par ailleurs très rapide. Aux Etats-Unis, l'Amiga devrait coûter environ 1 300 $ sans écran. ■

LOGICIELS

Forth
Cassette pour MO 5
Edité par Loriciels
Prix : 320 FF

CETTE version est du type Fig Forth 79 et comporte à ce titre toutes les fonctions de celui-ci. En supplément, le Forth MO 5 possède des instructions permettant de gérer le graphisme de manière efficace ainsi que la possibilité de créer des « lutins ». Le logiciel est présenté dans un coffret avec une petite notice d'une cinquantaine de pages.

L-Basic
Disquette pour C.64
Edité par Loriciels
Prix : 695 FF

LE classeur dans lequel est logée la disquette abrite un mode d'emploi réparti en onze chapitres (100 pages). Le logiciel est destiné à des développeurs, c'est-à-dire des programmeurs désireux de commercialiser leurs produits. L-Basic leur apporte de nouvelles fonctions, notamment dans le domaine du graphisme et des sons. Pourtant, un autre public paraît aussi visé : celui des débutants auxquels l'auteur, animateur de profession, à l'habitude de s'adresser.

Easy-Amscalc
Tableur
Cassette pour Amstrad
Edité par Amsoft
Prix : 245 FF

LE tableur, qui s'étend sur 36 lignes et 26 colonnes, permet la représentation graphique des calculs sous formes de diagrammes en bâtons.

Comme pour la plupart des logiciels de ce type, on ne peut pas utiliser Easy-Amscalc sans avoir auparavant consulté la notice qui l'accompagne (24 pages). Il est dommage de ne pas trouver à l'écran des menus détaillés indiquant chronologiquement les opérations à effectuer pour établir des légendes, entrer des données, les calculer puis les sauvegarder.

Cachez ce TO9...

ENFIN un vrai clavier chez **Thomson** ! Même si le constructeur ne nous a pas laissé voir la machine-à-faire-couler-de-l'encre, tout semble confirmer que le TO9 est doté d'un véritable clavier mécanique. ■

LOGICIELS

Forth
Microcassette pour Sinclair QL
Edité par Computer One
Distribué par Sinclair/Direco
Prix : 690 FF

CETTE version du Forth destiné au QL est au standard 83. Elle possède des extensions (le calcul en virgule flottante, le son et la gestion des fenêtres) qui permettent d'utiliser toutes les possibilités du QL. Elle dispose aussi d'un macro-assembleur et permet de travailler en « multitâches ». Un défaut de taille pour ce côté-ci de la Manche : la notice est en anglais.

Forth
Cassette pour Canon X-07
Edité par Logi'stick
Distribué par DDI
Prix : 240 FF

ENCORE un Forth ! Celui-ci est au standard 79. Son adaptation au X-07 a néanmoins conduit l'auteur à en modifier quelques points. Ceux-ci font l'objet d'un des chapitres du manuel qui accompagne la cassette. L'utilisateur, s'il ne connaît pas encore le langage, devra acquérir un ouvrage d'initiation au Forth. Le logiciel nécessite la possession d'un module 8 Ko portant la mémoire vive du X-07 à 16 Ko. Le manuel (environ 30 pages) est au format 21 × 29,7.

Le Basic
Auteur : Raynald Donais
Disquette pour Apple II 64 Ko
Edité par Puce
Distribué par Magnard Informatique
Prix : 990 FF

LES personnes que la langue anglaise horrifie et qui désirent se mettre au Basic pourront jeter leur dévolu sur ce logiciel. Conçu et édité par une équipe canadienne, il a été sélectionné par le distributeur français pour ses qualités pédagogiques. Chaque instruction du langage le plus utilisé parmi les non-professionnels trouve ici son équivalent en français. PRINT devient ECRIS, INPUT, DEMANDE, etc. Destiné à ceux qui veulent apprendre à programmer dans la langue de... Molière.

Basic étendu musical
Cassette pour C.64
Edité par Ere Informatique
Prix : 250 FF

LE logiciel permet de créer des mélodies composées de une à trois voix. Celles-ci peuvent être sauvegardées, modifiées et intégrées dans une liste Basic. Lors de leur exécution, les notes de la partition jouée défilent à l'écran sur trois colonnes.

UN LIVRE

Amstrad, le Basic au bout des doigts
Kampow
Edité par Micro Application
1985, 190 pages
Prix : 149 FF

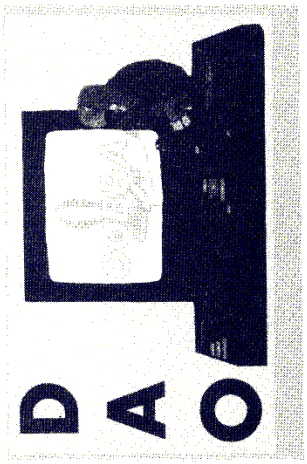
CES 190 pages à l'écriture serrée, traduites de l'allemand, sont visiblement destinées aux débutants. Elles sont d'un parfait classicisme après un premier chapitre qui évacue trop rapidement (une vingtaine de pages) tout ce qu'il faut savoir sur la structure des programmes : analyse (organigramme quand tu nous tiens !), les divers systèmes numériques, les bits et leurs compères les octets.

Une certaine recherche pédagogique montre le bout de son nez avec quelques exercices faciles, accompagnés de leur solution. Les chapitres suivants emmènent le lecteur d'INPUT à EVERY via WINDOW en faisant alterner pour chaque instruction étudiée les explications fonctionnelles, les programmes de démonstration et les applications avec corrigés.

Un reproche sur la qualité d'impression du livre : la composition, certainement due à une imprimante, rend la lecture malaisée. Pour le prix, on aurait espéré mieux.

Pour tirer profit des enseignements de l'ouvrage, il sera bon de le tenir d'une main tout en manipulant l'Amstrad de l'autre. La méthode ne brille peut-être pas par son originalité, mais elle a déjà fait ses preuves. C'est en tout cas ce qui fait la différence avec le manuel de l'utilisateur livré avec la machine. Ce dernier est une notice de références un peu trop aride et incomplète pour débuter, celui-ci est un outil d'apprentissage qui en vaut bien un autre. A destination de ceux qui veulent faire leurs premières armes sur l'ordinateur d'Amstrad. JPL ■

UNE CASSETTE



DAO
Logiciel graphique
Cassette pour Amstrad CPC 464/664
Edité par Cobra Soft
Prix : 120 FF

DAO, utilitaire, permet de créer à partir du clavier, des dessins qui peuvent être utilisés dans d'autres programmes. Le travail s'effectue très classiquement par l'intermédiaire du pavé numérique qui permet de déplacer dans huit directions un stylo (pixel) sur l'écran. Il y a trois modes graphiques avec le nombre de couleurs, correspondant, onze commandes pour les déplacements, le choix des couleurs, l'épaisseur des traits, etc. Les manipulations sont aisées mais les lignes se tracent lentement (le logiciel est écrit en Basic), et l'on aurait aimé une véritable « gomme » pour effacer facilement une portion du dessin en cours.

D'autres fonctions, plus élaborées, complètent la palette du créateur d'images lui permettant d'obtenir cercles, disques et rectangles et rendant possible le transfert de blocs graphiques (grâce à une routine en langage-machine). Si l'on ajoute à cela l'écriture de texte en un point quelconque de l'écran, l'affichage instantané des coordonnés du stylo, la mémorisation de points présélectionnés sans oublier la sauvegarde du dessin sur cassette, on aura fait le tour de ce qu'offre le logiciel.

DAO est donc relativement complet, simple d'emploi et les commandes sont bien pensées. Cette simplicité d'ensemble est d'autant plus appréciable que la notice d'utilisation est bien légère.

Il est dommage que les manipulations du dessin se limitent à la seule fonction de transfert de bloc (pas d'inversion, pas de zoom ni de superposition), mais pour le prix, on ne peut tout de même pas tout avoir ! JPL ■

Vente par correspondance

LE troisième catalogue **Moore Paragon** (« consommables » et accessoires informatiques) est disponible. Pour se le procurer, il suffit d'appeler le (16) 05 27 78 11 (numéro de téléphone vert). ■

TRIANGLES ET BONNES RÉSOLUTIONS

SANS aborder dans le détail les différents problèmes courants en topographie, examinons la pierre « angulaire » de ce type de questions : la résolution des triangles.

Toute figure de géométrie formée de lignes droites peut se décomposer en triangles. C'est de cette simple propriété que le triangle tire toute son importante.

Ainsi donc, quel que puisse être le problème de géométrie qui se pose, nous pourrons toujours le ramener à la résolution des divers triangles qui le composent pour peu qu'aucune courbe n'intervienne.

Théoriquement, le nombre de ces cas de résolution n'est pas limité. On pourra toujours en inventer un de plus. Par exemple : construire un triangle connaissant un côté, un angle et une hauteur ou une médiane ou (pourquoi pas ?) la moitié du rayon du cercle inscrit. Cependant, tous ces cas se ramènent à quatre cas classiques.

Ce programme pour ZX Spectrum

donne les solutions à ces quatre cas, c'est-à-dire connaissant :

a) les trois côtés ;
b) deux côtés et l'angle compris ;
c) deux côtés et un angle adjacent ;
d) un côté et deux angles.

On dispose ainsi d'un moyen de résoudre tous les problèmes de topographie plane qui peuvent se poser, et, par exemple :

- distance à un objet de hauteur connue ;
- distance à un objet de hauteur inconnue ;
- distance entre deux points inaccessibles.

De tous côtés, sous tous les angles

Le programme ci-après permet, de trouver les solutions à cette série (non limitative) de cas, et il peut être remanié pour traiter plus commodément l'un ou l'autre d'entre eux : il est quand même plus pratique de répondre à des INPUT qui s'expriment directement dans le langage du problème à traiter.

Pour utiliser le programme, il suffira de suivre les instructions portées sur l'écran pour chacun des quatre cas de résolution.

Il sera possible de connaître les trois angles, les trois côtés, les trois hauteurs,

TRIANGLES ET BONNES RÉSOLUTIONS

Résolution des triangles
Programme pour ZX Spectrum
Auteur Lucien Strebler
Copyright *LIST* et l'auteur

```
  78 LET w=0: CLS : PRINT AT 1,3
;"RESOLUTION DES TRIANGLES": LET
 lx=1: LET nlx=1: LET cx=3: LET
nx=24: LET ser=1: GO SUB 9500: L
ET ser=2: GO SUB 9500
  80 PRINT AT 4,0;"Definis par :
"
  82 PRINT AT 8,0;"1/ Les trois
cotes."
  84 PRINT AT 10,0;"2/ 2 cotes e
t l'angle compris."
  86 PRINT AT 12,0;"3/ 2 cotes e
t un angle adjacent."
  88 PRINT AT 14,0;"4/ 1 cote et
 2 angles adjacents."
  90 INPUT "Numero choisi ? ";no
  91 IF no=1 THEN GO TO 100
  92 IF no=2 THEN GO TO 200
  94 IF no=3 THEN GO TO 350
  96 IF no=4 THEN GO TO 400
  98 GO TO 90
 100 CLS : GO SUB 500: PRINT AT
20,11;"a";AT 10,13;"b";AT 11,2;"
c"
 110 INPUT "Cote a ? ";a: PRINT
AT 1,10;"a= ";a; INVERSE 1;AT 20
,11;"a"
 112 INPUT "Cote b ? ";b: PRINT
AT 2,10;"b= ";b; INVERSE 1;AT 10
,13;"b"
 114 INPUT "Cote c ? ";c: PRINT
AT 3,10;"c= ";c; INVERSE 1;AT 11
,2;"c"
 120 GO SUB 940: GO SUB 510
 126 LET n=ad: GO SUB 9550: LET
as=n: LET n=bd: GO SUB 9550: LET
 bs=n: LET n=cd: GO SUB 9550: LE
T cs=n
 128 PRINT AT 1,21;"A= ";INT (as
+100+.7)/100;AT 2,21;"B= ";INT (
bs+100+.7)/100;AT 3,21;"C= ";INT
 (cs+100+.7)/100: GO SUB 930
 130 PRINT AT 5,11;"Surface.. = "
;INT (s+10+.5)/10;AT 6,11;"Perim
etre= ";INT (20*p+.5)/10
 132 LET lx=5: LET nlx=2: LET cx
=11: LET nx=19: LET ser=2: GO SU
B 9500
 135 INPUT "Calcul des hauteurs
                 oui ou non ?(o/n) et
..COPY ?";r$
 140 IF r$="o" THEN GO TO 160
 145 IF r$="n" THEN GO TO 170
 150 IF r$="z" THEN COPY : GO TO
135
 155 GO TO 135
 160 LET v=0: GO SUB 570: GO SUB
524
 165 PRINT AT 8,20;"h1= ";INT (h
1+10+.5)/10;AT 9,20;"h2= ";INT (
h2+10+.5)/10;AT 10,20;"h3= ";INT
 (h3*10+.5)/10
 170 INPUT "Calcul des medianes
                 oui ou non ?(o/n) et
COPY..?";s$
 175 IF s$="o" THEN GO SUB 910:
GO TO 200
 180 IF s$="n" THEN GO SUB 910:
GO TO 210
 185 IF s$="z" OR s$="Z" THEN CO
PY : GO TO 170
 190 GO TO 170
 200 GO SUB 590: GO SUB 526
 205 PRINT AT 11,20;"m1= ";INT (
m1+10+.5)/10;AT 12,20;"m2= ";INT
 (m2+10+.5)/10;AT 13,20;"m3= ";I
NT (m3*10+.5)/10
 210 INPUT "Calcul des bissectri
ces          oui ou non ?(o/n) et
..COPY ?";t$
 215 IF t$="o" AND s$="n" THEN G
O SUB 590: GO TO 235
 216 IF t$="o" THEN GO TO 235
 220 IF t$="n" THEN GO TO 250
 225 IF t$="z" OR t$="Z" THEN CO
PY : GO TO 210
 230 GO TO 210
 235 PRINT AT 7,5;"b1";AT 12,10;
"b2";AT 17,13;"b3"
 240 GO SUB 532
 245 PRINT AT 14,20;"b1= ";INT (
b1+10+.5)/10;AT 15,20;"b2= ";INT
 (b2+10+.5)/10;AT 16,20;"b3= ";I
NT (b3*10+.5)/10
 250 INPUT "Cercle circonscrit R
,inscrit r oui ou non (o/n) et..
COPY ?";r$
 253 IF r$="o" THEN GO TO 260
 255 IF r$="n" AND v=0 THEN RUN
 256 IF r$="n" THEN GO TO 265
 257 IF r$="z" OR r$="Z" THEN CO
PY : GO TO 250
 258 GO TO 250
 260 GO SUB 538: PRINT AT 17,21;
"R= ";INT (rc*10+.5)/10;AT 18,21
;"r= ";INT (ri*10+.5)/10: INPUT
"COPY ? ";r$: IF r$="Z" OR r$="Z
" OR r$="o" THEN COPY
 261 IF v=0 THEN RUN
 265 PRINT AT 8,3;"                    "
;AT 9,3;"  PRESSEZ  ";AT 10,3;"
UNE TOUCHE ";AT 11,3;"

 266 IF INKEY$<>"" THEN GO TO 39
6
 267 GO TO 266
 300 CLS : GO SUB 500: PRINT AT
10,13;"b";AT 11,2;"c"
 305 PLOT 36,150: DRAW 13,3,.8
 310 INPUT "Cote b ? ";b: PRINT
AT 1,10;"b= ";b; INVERSE 1;AT 10
,13;"b"
 315 INPUT "Cote c ? ";c: PRINT
AT 2,10;"c= ";c; INVERSE 1;AT 11
,2;"c"
 320 INPUT "ANGLE A ? ";as: PRIN
T AT 3,10;"A= ";as; INVERSE 1;AT
 0,4;"A": GO SUB 940
 325 LET n=as: GO SUB 9560: LET
ad=n: LET a=SQR (b↑2+c↑2-2*b*c*C
OS (ad*PI/180))
 330 GO SUB 510: PRINT AT 1,21;"
a= ";INT (a*10+.5)/10
 335 LET n=bd: GO SUB 9550: LET
bs=n: PRINT AT 2,21;"B= ";INT (b
s*100+.7)/100
 340 LET n=cd: GO SUB 9550: LET
cs=n: PRINT AT 3,21;"C= ";INT (c
s*100+.7)/100
 345 GO SUB 930: GO TO 130
 350 CLS : GO SUB 500: PRINT AT
20,11;"a";AT 10,13;"b"
 355 PLOT 11,19: DRAW 8,-10,-1.4
 360 INPUT "Cote a ? ";a: PRINT
AT 1,10;"a= ";a; INVERSE 1;AT 20
,11;"a"
 365 INPUT "Cote b ? ";b: PRINT
AT 2,10;"b= ";b; INVERSE 1;AT 10
,13;"b"
 370 INPUT "ANGLE B ? ";bs: PRIN
T AT 3,10;"B= ";bs; INVERSE 1;AT
 20,0;"B": GO SUB 940
 374 IF a>b THEN LET w=1
 375 LET n=bs: GO SUB 9560: LET
bd=n: LET sin=a/b*SIN (bd*PI/180
): IF sin>1 THEN GO TO 950
 376 LET ad=ASN sin*180/PI: GO S
UB 945
 395 GO SUB 930: GO TO 130
 396 CLS : PRINT INVERSE 1;AT 10
,1;"IL Y A UNE DEUXIEME SOLUTION
": PAUSE 100: CLS : GO SUB 500:
PRINT INVERSE 1;AT 20,11;"a";AT
10,13;"b";AT 20,0;"B": PRINT AT
1,10;"a= ";a;AT 2,10;"b= ";b;AT
3,10;"B= ";bs
 397 GO SUB 940: PLOT 11,19: DRA
W 8,-10,-1.4
 398 LET ad=180-ad: GO SUB 945:
GO SUB 930: LET w=0: GO TO 130
 400 CLS : GO SUB 500: PRINT AT
20,11;"a"
 405 PLOT 11,19: DRAW 8,-10,-1.4
: PLOT 162,9: DRAW 5,10,-.9: PLO
T 161,9: DRAW 5,10,-.9
 410 INPUT "Cote a ? ";a: PRINT
AT 1,10;"a= ";a; INVERSE 1;AT 20
,11;"a"
 415 INPUT "ANGLE B ? ";bs: PRIN
T AT 2,10;"B= ";bs; INVERSE 1;AT
 20,0;"B"
 420 INPUT "ANGLE C ? ";cs: PRIN
T AT 3,10;"C= ";cs; INVERSE 1;AT
 20,22;"C": GO SUB 940
 425 LET n=bs: GO SUB 9560: LET
bd=n: LET n=cs: GO SUB 9560: LET
 cd=n
 430 LET ad=180-bd-cd: LET n=ad:
GO SUB 9550: LET as=n
 435 LET b=a*SIN (bd*PI/180)/SIN
 (ad*PI/180): LET c=a*SIN (cd*PI
/180)/SIN (ad*PI/180)
 440 GO SUB 510: PRINT AT 1,21;"
```

```
A= ";INT (as*100+.7)/100;AT 2,21
;"b= ";INT (b*10+.5)/10;AT 3,21;
"c= ";INT (c*10+.5)/10
 445 GO SUB 930: GO TO 130
 500 PLOT 9,9: DRAW 30,158: DRAW
 136,-158: DRAW -166,0: PRINT AT
 20,0;"B";AT 20,22;"C";AT 0,4;"A
": RETURN
 510 LET p=(a+b+c)/2: IF (p-a)+(
p-b)+(p-c)<0 THEN GO TO 950
 511 LET s=SQR (p*(p-a)*(p-b)*(p
-c))
 512 LET sin=2*s/b/c: IF sin>1 T
HEN LET sin=1
 513 LET ad=ASN sin*180/PI
 514 LET sin=2*s/a/c: IF sin>1 T
HEN LET sin=1
 515 LET bd=ASN sin*180/PI
 516 LET sin=2*s/a/b: IF sin>1 T
HEN LET sin=1
 517 LET cd=ASN sin*180/PI
 518 IF at2>bt2+ct2 THEN LET ad=
180-ad
 520 IF bt2>at2+ct2 THEN LET bd=
180-bd
 522 IF ct2>at2+bt2 THEN LET cd=
180-cd
 523 RETURN
 524 LET h1=2/a*s: LET h2=2/b*s:
 LET h3=2/c*s: RETURN
 526 LET m1=SQR (2*(bt2+ct2)-at2
)/2
 528 LET m2=SQR (2*(at2+ct2)-bt2
)/2
 530 LET m3=SQR (2*(at2+bt2)-ct2
)/2: RETURN
 532 LET b1=2*SQR (b*c*p*(p-a))/
(b+c)
 534 LET b2=2*SQR (a*c*p*(p-b))/
(a+c)
 536 LET b3=2*SQR (a*b*p*(p-c))/
(a+b): RETURN
 538 LET rc=a*b*c/4/s: LET ri=s/
(1/TAN (bd*PI/360)+1/TAN (cd*PI/
360)): RETURN
 570 FOR i=166 TO 9 STEP -2: PLO
T INVERSE 0;40,i: NEXT i
 571 PLOT INVERSE 0;44,10: DRAW
INVERSE 0;0,3: DRAW INVERSE 0;-4
,0
 575 FOR i=9 TO 97 STEP 2: PLOT
INVERSE 0;i,i: NEXT i
 576 PLOT INVERSE 0;95,95: DRAW
INVERSE 0;-3,3: DRAW INVERSE 0;3
,3
 580 FOR i=175 TO 17 STEP -2: PL
OT INVERSE 0;i,9+(175-i)/4.3: NE
XT i
 581 PLOT INVERSE 0;19,46: DRAW
INVERSE 0;1,3: DRAW INVERSE 0;-3
,1
 585 PRINT AT 7,5;"h1";AT 12,10;
"h2";AT 17,13;"h3"
 586 RETURN
 590 FOR i=166 TO 9 STEP -2: PLO
T 91-i/3.2,i: NEXT i
 595 FOR i=10 TO 105 STEP 2: PLO
T i,i/1.2+2: NEXT i
 600 FOR i=175 TO 24 STEP -2: PL
OT i,9+(175-i)/1.95: NEXT i
 605 PRINT AT 7,5;"m1";AT 12,10;
"m2";AT 17,13;"m3"
 606 RETURN
 910 IF r$="o" THEN LET o=1: GO
SUB 570: PRINT AT 7,5;"  ";AT 12
,10;"  ";AT 17,13;"  ": RETURN
 930 LET lx=1: LET nlx=3: LET cx
=21: LET nx=9: LET ser=2: GO SUB
9500
 940 LET lx=1: LET nlx=3: LET cx
=10: LET nx=9: LET ser=2: GO SUB
9500: LET ser=3: GO SUB 9500: R
ETURN
 945 LET n=ad: GO SUB 9550: LET
as=n
 946 LET cd=180-bd-ad
 947 LET n=cd: GO SUB 9550: LET
cs=n: LET c=SQR (at2+bt2-2*a*b*C
OS (cd*PI/180))
 948 GO SUB 510: PRINT AT 1,21;"
c= ";INT (c*10+.5)/10;AT 2,21;"A
= ";INT (as*100+.7)/100;AT 3,21;
"C= ";INT (cs*100+.7)/100: RETUR
N
 950 CLS : PRINT INVERSE 1;AT 10
,5;"TRIANGLE IMPOSSIBLE": PAUSE
100: CLS : RUN
9500 PLOT cx+8-ser-1,175-8+lx+se
r: DRAW nx+8+ser+2+1,0: DRAW 0,1
-nlx+8-ser+2: DRAW -nx+8-ser+2-1
,0: DRAW 0,nlx+8+ser+2-1: RETURN

9550 LET z=SGN n: LET mm=(ABS n-
INT ABS n)*60: LET ss=(mm-INT mm
)*60
9551 LET n=(INT ABS n+INT mm/100
+INT (ss+.5)/10000)*z: RETURN
9560 LET z=SGN n: LET n=ABS n: L
ET e=(n-INT n)*100: LET n=INT n+
INT (e+.01)/60+(e-INT (e+.01))/3
6: LET n=n*z: RETURN
```

Le programme dans ses grandes lignes

Lignes 10 à 34 : menu
Lignes 100 à 128 : triangle défini par a b c
Lignes 130 à 132 : calcul surface et périmètre
Lignes 135 à 165 : traitement des hauteurs
Lignes 170 à 205 : traitement des médianes
Lignes 210 à 245 : traitement des bissectrices
Lignes 250 à 260 : traitement des cercles inscrits, circonscrits
Lignes 300 à 345 : triangle défini par b c a
Lignes 350 à 395 : triangle défini par a b B
Lignes 400 à 455 : triangle défini par a B C

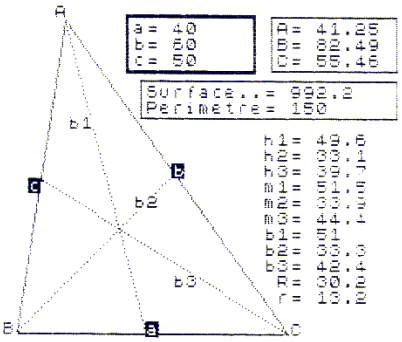
Sous-programmes

Ligne 500 : trace le triangle de base
Ligne 510 : calcule p s A B C
Ligne 524 : calcule les 3 hauteurs
Ligne 526 : calcule les 3 médianes
Ligne 532 : calcule les 3 bissectrices
Ligne 538 : calcule les rayons des cercles inscrits, circonscrits
Ligne 570 : tracé et effacement des hauteurs
Ligne 590 : tracé des médianes
Ligne 910 : effacement des hauteurs
Ligne 930 : encadrement des premiers résultats
Ligne 940 : encadrement des entrées
Ligne 950 : affiche les cas d'impossibilité
Ligne 9500 : sous-programme d'encadrement général
Ligne 9550 : transformation en dms
Ligne 9560 : transformation en dec

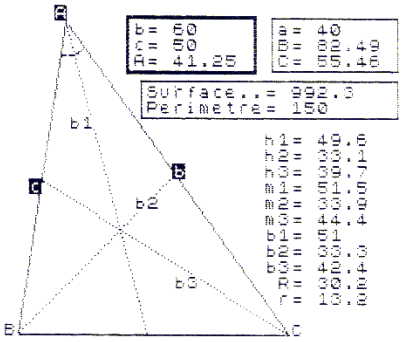
Définition des variables

a, b, c: côtés du triangle
as, bs, cs : angles au sommet sexagésimaux
ad, bd, cd : angles au sommet décimaux
s : surface
P : demi-périmètre
r$, s$, t$: chaînes de caractères des INPUT à options
o : oui
n : non
z : pour COPY et calculs dms/dec
mm, ss, e : pour calculs dms/dec
h1, h2, h3 : hauteurs
m1, m2, m3 : médianes
b1, b2, b3 : bissectrices
re : rayon du cercle circonscrit
ri : rayon du cercle inscrit

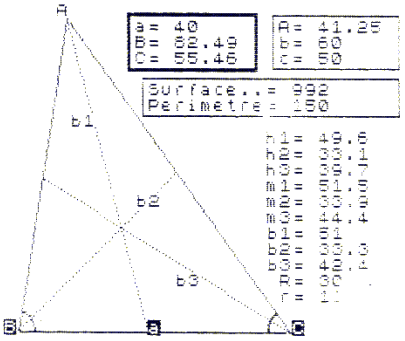
TRIANGLE DEFINI PAR 3 COTES



TRIANGLE DEFINI PAR 2 COTES ET L'ANGLE COMPRIS



TRIANGLE DEFINI PAR UN COTE ET LES ANGLES ADJACENTS



les trois bissectrices, les trois médianes, les rayons des cercles circonscrits et inscrits, le périmètre et la surface. De quoi rêver en terminale !

On trouvera ci-dessus, illustrée grâce à trois copies d'écran, les solutions correspondant aux cas d'un triangle défini par ses trois côtés, d'un triangle défini par deux côtés et l'angle compris, et enfin d'un triangle défini par un côté et les deux angles adjacents.

Les données du problème apparaissent dans le premier encadré, les résultats principaux dans les deux autres encadrés et les résultats secondaires en vrac.

Lucien STREBLER

LES JEUX ET CASSE-TÊTE INFORMATIQUES

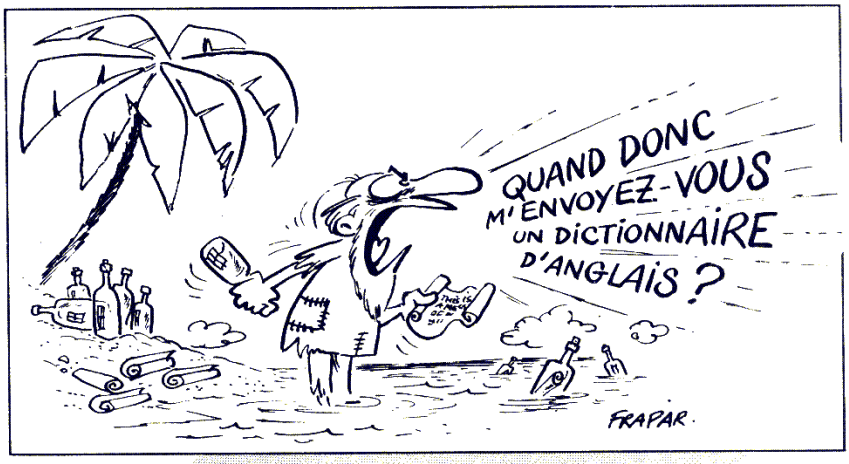
de Thierry CHAMORET

LES jeux et casse-tête qui vous sont proposés dans cette rubrique ont plusieurs aspects. Tout d'abord, ils peuvent être pris sous l'angle ludique, c'est-à-dire qu'il s'agit de jeux, de petits problèmes plus ou moins faciles à résoudre. Ils ont également un aspect pratique. Ils permettent en effet à chacun d'exercer son agilité logique. Et il n'est pas nécessaire, pour trouver la solution, d'avoir un ordinateur sous la main...

39

Messages d'erreur

Les messages d'erreur qui apparaissent à l'écran ne sont pas toujours faciles à comprendre, surtout s'ils sont en anglais. Pouvez-vous faire correspondre, à chaque numéro de la liste des messages anglais, la lettre de son équivalent français ?



Messages anglais

1. Non dimensioned variable has subscript
2. Invalid variable bounds
3. One or more invalid chars skipped
4. Non dummy has variable dimension
5. Substring out of range
6. Mismatched parentheses
7. Unnumbered statement follows transfer

8. Transferts into do loop
9. Improper parameter matching
10. Subscript out of range

Equivalents français

A. Bornes d'une sous-chaîne de caractères non compatibles avec la dimension, ou sous-chaîne trop longue

B. Une variable déclarée comme simple est utilisée avec un indice

C. Limites d'une variable invalides (dimensionnement erroné d'un tableau)

D. Un tableau est déclaré avec un nombre d'éléments variable

E. Paramétrage incorrect (la liste des paramètres d'un sous-programme est incompatible avec la déclaration)

F. Une instruction GOTO effectue un branchement à l'intérieur d'une boucle

G. Une instruction non étiquetée suit un GOTO (elle ne peut donc jamais être exécutée)

H. Un ou plusieurs caractères ne sont pas reconnus et sont sautés

I. Indice non compatible avec la dimension d'un tableau

J. Le nombre de parenthèses gauches est différent du nombre de parenthèses droites

40

Optimisation

On considère le programme suivant :

```
10 FOR I=N1 TO N2
20     A(I)=2*I+3
30     FOR J=1 TO 1000
40       FOR K=1 TO 1000
50         B(J,K)=0
60       NEXT K
70     NEXT J
80 NEXT I
```

Le temps de calcul de ce programme n'est pas négligeable. En effet, les boucles des lignes 30 et 40 allant de 1 à 1000, la ligne 50 est exécutée 1000000*(N2 – N1 + 1) fois. Il s'avère alors nécessaire d'optimiser ce programme qui initialise les tableaux A et B. On a donc pensé le remplacer par :

```
10 FOR I=N1 TO N2
20     A(I)=2*I+3
30 NEXT I
40 FOR J=1 TO 1000
50     FOR K=1 TO 1000
60       B(J,K)=0
70     NEXT K
80 NEXT J
```

Mais ce programme devient faux. Pourquoi ?

41

Trouver des formules

L'opérateur MOD calcule le reste de la division entière. Ainsi, I MOD 2 vaut 0 si I est divisible par 2, et 1 dans le cas contraire. Donc, la ligne de programme : IF (I MOD 2)=0 THEN X=A ELSE X=B, affecte la valeur de A à X si I est pair, et la valeur de B si I est impair. Cette ligne peut être remplacée par une simple expression arithmétique. L'objet de ce problème consiste alors à trouver le plus grand nombre d'expressions qui remplacent cette ligne. Pour notre part, nous en avons trouvé quatre.

Toutes les opérations ou fonctions disponibles sur votre ordinateur peuvent, bien entendu, être utilisées.

Les solutions de ces jeux se trouvent à la page 78 de ce numéro

42

Les deux plus grands

Si le problème de la recherche du plus grand élément d'un tableau est résolu, celui de la recherche des deux plus grands éléments peut encore être soulevé. Le programme suivant semble proposer une solution :

```
10 M1=1
20 FOR I=2 TO N
30     IF T(M1)<T(I) THEN M1=I
40 NEXT I
50 M2=1
60 FOR I=2 TO N
70     IF T(M2)<T(I) AND T(M1)
       <>T(I) THEN M2=I
80 NEXT I
90 PRINT T(M1),T(M2)
```

Ce programme est composé de deux boucles qui permettent de rechercher respectivement le premier et le second plus grand élément du tableau T. Une recherche simultanée de ces deux éléments peut être faite avec une seule boucle :

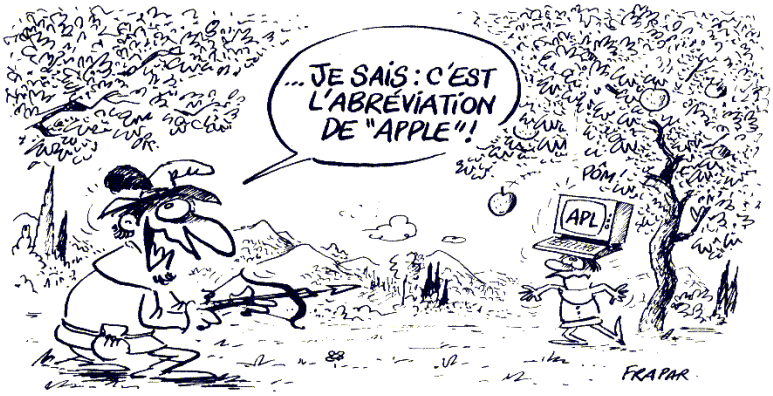
```
10 M1=1
20 M2=2
30 FOR I=3 TO N
40     IF T(M1)<=T(I) THEN M1
       =1: GOTO 60
50     IF T(M2)<T(I) THEN M2=I
60 NEXT I
70 PRINT T(M1),T(M2)
```

Ce dernier programme est plus court et plus rapide que le précédent. Toutefois, rien ne sert d'améliorer un programme qui ne fonctionne pas. En effet, si les résultats attendus sont le plus souvent justes, ils sont faux dans certaines circonstances. Comment corriger chacune de ces deux versions ?

LES JEUX ET CASSE-TÊTE INFORMATIQUES

43

Langages



La liste suivante est constituée de dix noms de langages informatiques. Pour chacun d'entre eux, pouvez-vous indiquer la signification de l'abréviation utilisée ?

1. APL
2. Basic
3. Cobol
4. Fortran
5. GAP
6. Lisp
7. LSE
8. Pascal
9. PL/1
10. Prolog

44

Tous les sens

Selon les langages, une même instruction peut avoir différents sens. Par exemple, les résultats de l'expression A = B = 0 varient selon les langages. Ainsi, on trouve pour la variable A :
0 avec le langage L
1 avec le langage M
2 avec le langage N

Compte tenu de ces résultats, pouvez-vous expliquer comment agit chacun de ces langages ?

45

Racine numérique

La racine numérique d'un nombre (ou chiffre résiduel) est le chiffre qui résulte de la somme des chiffres qui composent le nombre, somme effectuée autant de fois qu'il est nécessaire pour arriver à une valeur sur un chiffre.

Par exemple, la racine numérique R de 156 s'obtient de la manière suivante :

- 156 donne 1 + 5 + 6 = 12;
- 12 donne 1 + 2 = 3.

Donc R = 3. Trouver la fonction la plus courte possible, à programmer en Basic ou en Pascal, qui retourne la racine numérique d'un nombre.

46

Rapports ambigus

Un moyen d'améliorer les performances d'un programme est de remplacer les divisions par des multiplications, autant qu'il est possible. En effet, si en une seconde l'ordinateur effectue 575 multiplications, dans le même temps, il ne traite que 312 divisions. Une économie de plus de 45 % de temps est donc envisageable.

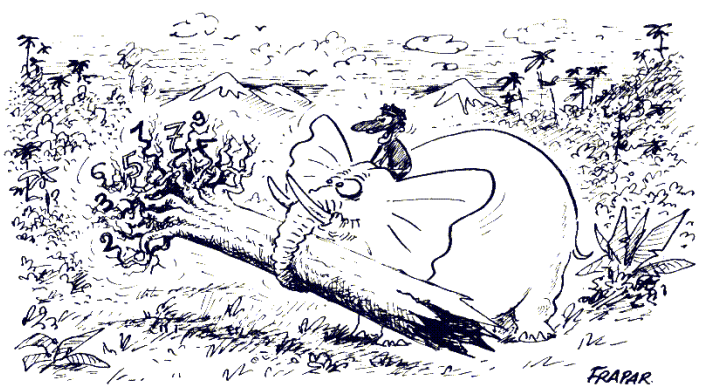
Sachant cela, un programmeur astucieux remplaça, dans un programme, la comparaison de deux fractions par celle de deux produits. Le test IF (A/B) > (C/D) THEN ... devint donc IF (A*D) > (C*B) THEN ... où A, B, C et D sont des variables réelles strictement positives.

Un rapide calcul montre que ces tests sont équivalents. Et pourtant une erreur s'est déclenchée à l'exécution du programme ainsi modifié. Quel est le libellé de cette erreur ?

47

Les surprises du calendrier

Le 31 octobre, veille de la Toussaint, n'est pas le 25 décembre, jour de Noël. Mais pouvez-vous montrer que :
31 OCT = 25 DEC



SOLUTIONS DU NUMÉRO PRÉCÉDENT

LES solutions présentées ici répondent aux problèmes posés dans le précédent numéro de LIST. Ce ne sont pas forcément les meilleures !

34

Rotations

Pour qu'un sous-programme chargé d'effectuer une rotation de pas P, dans un tableau T de longueur L soit le plus rapide possible, il est nécessaire de réduire la valeur du pas P. Ainsi, le premier indice du tableau étant égal à 1, une rotation de pas L+1 est équivalente à une rotation de pas 1. En outre, toutes les rotations de pas égal à 0, L, 2*L, ... ne doivent pas être traitées, ces pas représentant l'absence de rotation.

Enfin, il faut songer aux valeurs de pas négatives. Dans ce cas, on ajoute au pas la valeur L, autant de fois que nécessaire pour qu'il devienne positif. L'extrait de programme ci-dessous effectue la rotation des valeurs du tableau T le plus rapidement possible. Il accepte toutes les valeurs entières du pas P. Le résultat est donné dans le tableau U. L'opérateur DIV renvoie le résultat entier de la division appliquée aux deux nombres qui l'entourent.

```
P=P-L*(P DIV L)
IF P<0 THEN P=P+L
FOR I=1 TO L
  U((I+P-1)-L*((I+P-1) DIV L)
+1)=T(I)
NEXT I
```

35

Le langage du programmeur

Le programmeur un peu badin, qui appelle fréquemment une routine baptisée GIRL, programme en Fortran. Dans ce langage, les sous-programmes sont déclarés par SUBROUTINE et activés par CALL. Notre programmeur déclenche donc le traitement de la routine par l'instruction CALL GIRL !

36

Opérations prioritaires

Avec un ordinateur effectuant les opérations sur 16 bits, l'égalité A*(B/C)=(A*B)/C n'est pas toujours vérifiée. C'est le cas en particulier, lorsque les valeurs de A et B sont telles que leur produit dépasse 32767. En effet, pour l'évaluation de A*(B/C), le calcul de B/C est effectué en premier, grâce aux parenthèses. Le résultat de cette division est un nombre réel qui sera alors multiplié par A.

Lors du calcul de (A*B)/C, la multiplication est effectuée en priorité. Son résultat entier n'est pas converti sous forme réelle et, s'il dépasse 32767 (c'est-à-dire 2^15-1), un débordement se produit. Il n'est généralement pas signalé par les ordinateurs. Le résultat est alors faux et, après la division par C, on ne retrouvera pas le même résultat que pour la première expression.

Par exemple, avec A=200, B=300 et C=1, on a :
A*(B/C)=60000.0 et (A*B)/C=−5536.0 (−5536.0 est le résultat de 6000−65536, 65536 étant égal à 2^16).

37

Dans un fichier d'adresses

Si l'on décide de remplacer toutes les occurrences de "et" par "&" dans un fichier d'adresses, un intitulé tel que "M. et Mme Peter Maigret" devient "M. & Mme P&er Maigr&". Ce qui risque de surprendre le destinataire. Pour éviter cette erreur, il faut plutôt remplacer la chaîne " et " (un espace-et-un espace) par la chaîne " & " (un espace-&-un espace).

38

Arrondir des entiers

Un entier N peut être arrondi sur C chiffres avec quelques lignes de Basic :

```
10 N=10*N
20 N=INT(N/10^C)
30 N=N+5
40 N=INT(N/10)
50 N=N*N10^C
```

En prenant N=18392, on trouve bien que ce nombre arrondi sur deux chiffres vaut 18400 ; arrondi sur trois chiffres, il vaut 18000.

UN ÉDITEUR DE DESSINS POUR DAI

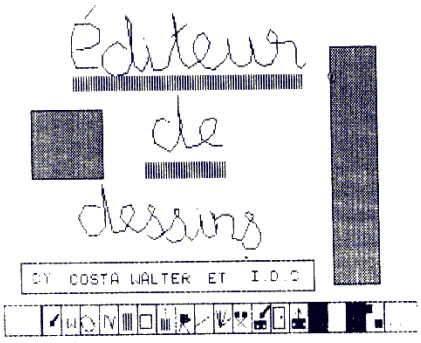
LE Dai, c'est connu, est apprécié pour ses qualités graphiques. Depuis son Basic, il est relativement aisé de confectionner de jolies images. Mais il faut dire que cela devient rapidement fastidieux, en raison de l'abondance de lignes de commandes à écrire. C'est là qu'intervient Éditeur de dessins. Ce logiciel dessine en temps réel au moyen d'une manette de jeux ou, mieux, d'une souris, et conserve en fichier l'œuvre ainsi créée.

Le logiciel *Éditeur de dessins* se présente sous forme de cassette audio, ou micro-cassette numérique, et se charge comme du langage-machine (UT R). Hélas, son système de protection emploie une procédure inélégante et gênante pour l'utilisateur. Sans « vendre la mèche », je puis vous dire que cette protection consiste à charger un fichier énorme, qui va de l'adresse 0 jusqu'à l'aire du STACK du processeur. Le démarrage automatique utilise le détournement d'un vecteur qui aboutit normalement à un message d'erreur. En conséquence, le chargement est très long (pas loin d'un quart d'heure), et moins fiable qu'une procédure normale. C'est dommage, car le logiciel proprement dit est plein de qualités.

J'ai essayé *Éditeur de dessins* avec la souris distribuée par le club IDC. Dans ces conditions, le clavier n'est pratiquement pas employé, et c'est très agréable. Au bas de l'écran, un bandeau contient les icônes mnémotechniques des différentes fonctions disponibles. Il suffit de pointer une icône avec la flèche qui symbolise l'emplacement du curseur. Ensuite, on « clique » la souris, et la commande est active.

Par exemple, pour écrire du texte, à un emplacement quelconque du dessin, il faut pointer la case contenant la lettre « W ». Dès ce moment, toute lettre frappée au clavier remplace à l'écran la flèche du curseur.

En appuyant sur la touche verte (ou grise) portant le symbole « flèche en haut » (en haut à gauche du clavier), du texte peut être tapé : il s'inscrit à partir de la position courante du curseur. C'est



La page de présentation du logiciel : écriture cursive et caractères d'imprimerie

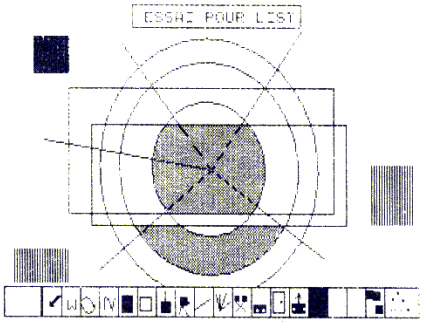
à ce moment qu'intervient la souplesse d'emploi d'*Éditeur de dessins* : ce texte a pu être formé n'importe où, dans l'écran, car il n'est pas encore incrusté. En frappant la touche RETURN, l'ensemble du texte devient le curseur, et peut ainsi être déplacé avec la souris, n'importe où dans la page graphique, sans écraser ce qui s'y trouve déjà. C'est en cliquant la souris que l'on incruste le texte à l'endroit désiré, et autant de fois que l'on veut, où l'on veut.

Un petit air de Macintosh

De plus, *Éditeur de dessins* comporte un jeu de caractères redéfinissables au gré de l'utilisateur (commande « T »). Le symbole choisi est amené dans le rectangle situé en bas à gauche de la page de redéfinition, et la modification se fait avec les touches de déplacement du curseur. On peut ainsi fabriquer n'importe quel symbole ou lettre, voire de petits dessins (un petit bonhomme et un chien font déjà partie du jeu de base).

Hormis ces préparatifs, la phase de réalisation du dessin se résume à cliquer une fonction et à la « coller » dans la page graphique, à l'endroit désiré. Il est ainsi facile de créer une série de cercles concentriques : on choisit le centre, en pointant l'icône POSITION (la flèche, dans le bandeau de commande), puis on clique le curseur à l'endroit voulu dans le dessin.

Ensuite, il suffit de choisir la fonction CERCLE, et d'aller pointer un endroit de la circonférence désirée pour obtenir son tracé. Même philosophie pour faire des « boîtes » vides ou pleines (BOX, et FILL) : après sélection de la fonction, on clique la position de deux coins opposés, et c'est tout.



Exemple de dessin : c'est la copie sur imprimante graphique qui a déformé les cercles

La redéfinition du jeu de caractères

Parmi les autres options se trouvent encore les classiques que sont les tracés de droites, de rayons concentriques. Le quatrième icône, en partant de la gauche sur le bandeau de commande, permet le dessin « à main levée » : le programme produit une trace qui suit fidèlement l'évolution de la souris. On peut ainsi écrire du texte en cursive, quasiment comme on le ferait avec un crayon. L'effet est surprenant.

Choisissez vos couleurs

Éditeur de dessins possède également des fonctions imitant l'aérographe : l'utilisateur peut choisir neuf tailles de points à inscrire dans son travail, et chaque point peut associer deux couleurs, pour obtenir des nuances. Cela fera peut-être moins regretter l'obligation de se limiter à quatre couleurs (choisies parmi seize), d'autant que l'icône n° 7 (le « pinceau ») donne la fonction PAINT, avec possibilité de tramage en deux couleurs. Dans ce cas, la seule restriction importante est la nécessité de peindre une surface *fermée*, et limitée par un trait d'une couleur différente de celles employées pour la peinture.

Si la surface est imparfaitement fermée (un pixel ouvert suffit), le coloriage déborde, et envahit... tout l'écran ! Frayeur garantie, surtout après de longues heures de labeur, mais, pas de panique : le mode d'emploi dit ce qu'il faut faire dans ce cas. On appuie sur la touche BREAK, qui ramène au menu initial, et on tape « E », pour faire redémarrer le programme, sans perdre le dessin.

Imaginons maintenant que ce dessin présente une erreur manifeste de conception. Il faut le corriger, et le programme ne s'intitule pas *Éditeur* pour rien. En effet, toutes les phases de réalisation du dessin sont mémorisées dans un fichier numérique. En cliquant la onzième fonction (les « ciseaux »), on passe dans le mode d'édition : le dessin se reconstruit pas à pas, en donnant des indications (dans les quatre lignes de texte, tout en bas de l'écran) : commande active, couleur, coordonnées, occupation du fichier. On peut alors corriger ce qui ne va pas, insérer d'autres éléments de dessin, ou encore en effacer, si l'on juge telle ou telle partie inesthétique ou inappropriée. Ainsi, toute latitude est laissée à « l'artiste » pour peaufiner son chef-d'œuvre.

Bien entendu, le travail achevé peut être sauvé sur mémoire de masse et rappelé ensuite. Il existe même une commande (MERGE) permettant la fusion de plusieurs fichiers picturaux. L'utilisateur peut ainsi confectionner des fragments de graphisme et les assembler selon ses besoins. Détail intéressant, le fichier numérique codant le dessin est beaucoup moins long à charger qu'une sauvegarde d'écran graphique haute résolution, et il est certainement possible de le rappeler depuis un programme Basic (à condition d'avoir percé les secrets de son codage).

Pour ce qui est de l'apprentissage, il faut bien avouer que la présence du bandeau d'icônes facilite grandement la maîtrise du logiciel, et la manipulation de la souris devient très vite naturelle. Le concept du « cliquer-coller » se marie particulièrement bien avec un éditeur de dessins, et c'est le cas ici.

Voici en définitive un logiciel que tous les Daïstes amateurs de graphisme se doivent au moins d'essayer, car il est à la fois simple et puissant.

Alain MARIATTE

Le logiciel en quelques lignes

Nom : *Éditeur de dessin*
Ordinateur : *Dai PC et Dai T*
Forme : *cassette audio ou micro-cassette digitale*
Distribué par : *International Dai Club (IDC)*
• Fabrice Duluins, 4, allée Tour Renard, B-1400 Nivelles, Belgique, ou
• Bruno Delannay, résidence « Les Acacias », bât. B3, avenue de Saige, 33600 Pessac, France
Prix public : *195 FF (cassette audio), 218 FF (cassette digitale)*
Principales orientations : *création de dessins en haute résolution en temps réel, au moyen d'une manette de jeux ou d'une souris. Graphisme et texte mélangeables sans restriction.*

ADRESSAGE IMMÉDIAT, ADRESSAGE ÉTENDU

Un chapitre essentiel de la programmation en Assembleur est celui qui traite des modes d'adressage. Le microprocesseur 6809 en propose neuf qui constituent une des principales raisons de sa puissance logicielle. L'étude de deux de ces modes, immédiat et étendu, met en évidence leur importance.

Choisir un mode d'adressage revient à choisir la façon selon laquelle une valeur est chargée à une adresse ou dans un registre. Le mode le plus simple est le mode immédiat : un registre du microprocesseur est chargé avec une valeur.

Par exemple, l'instruction LDA #$1A contient tous les renseignements de l'adressage immédiat. Les deux premières lettres, LD, sont l'abréviation de *load*, charger. Le premier A représente l'accumulateur A. Le signe # *(cross-hatch)* caractérise l'adressage immédiat, le signe $, l'hexadécimal. Et 1A est la valeur hexadécimale à charger. Ainsi, l'instruction LDA #$1A signifie : charger, en mode immédiat, la valeur hexadécimale 1A dans le registre A.

Si l'on travaille sur l'accumulateur D (qui dans ce cas est sur 16 bits puisqu'il résulte de la concaténation des deux accumulateurs A et B, chacun de 8 bits), on écrira : LDD #$20A2. Cette fois, on charge la valeur hexadécimale 20A2, représentée sur 16 bits (8354 en décimal).

On peut également charger en mode immédiat les registres d'index 16 bits, X et Y, et les piles système et utilisateur, S et U. Les codes mnémoniques sont alors : LDX, LDY, LDS et LDU.

D'autres instructions du 6809 fonctionnent en mode immédiat. C'est le cas de EXG, l'instruction qui permet d'échanger les valeurs de deux registres (le SWAP du Basic en quelque sorte). Par exemple, la suite d'instructions :

```
LDA #$1A
LDB #$20
EXG A,B
```

échange les contenus de A (ici 1A en hexa) et de B (20 en hexa). Le code-objet de cette suite d'instructions est :

86 1A (86 pour LDA en mode immédiat)
C6 20 (C6 pour LDB en mode immédiat)
1E 89 (1E pour EXG, 8 pour A et 9 pour B).

Un bon endroit pour lire

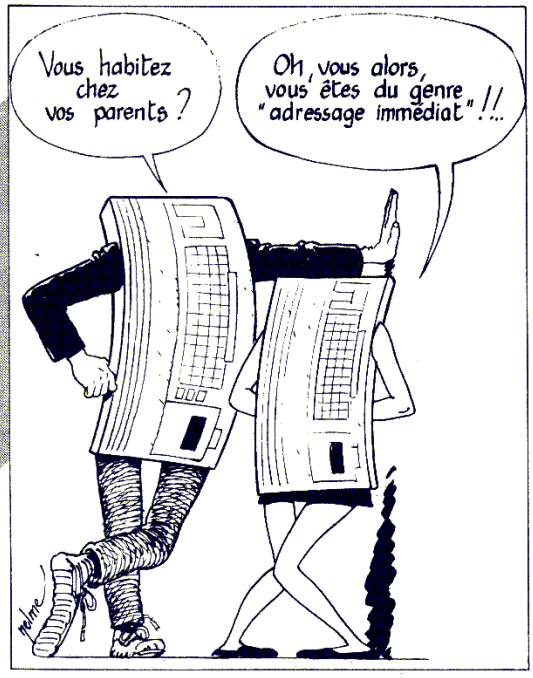
Comme il ne nous est pas possible de lire directement les valeurs contenues dans A et B, il faut commencer par les transférer à des adresses mémoire qui pourront être lues en Basic avec PEEK(ADR). Les instructions de stockage, STA et STB, doivent donc être suivies de l'adresse où l'on veut stocker.

```
10 CLEAR,&H9FFF
20 FOR I=1 TO 13
30 READ A$:A$="&H"+A$:A=VAL(A$)
40 POKE &HA000+I,A
50 NEXT I
60 EXEC &HA000
70 PRINT "JE TROUVE, EN &HDFFE, LA VALEUR ";
   HEX$(PEEK(&HDFFE))
80 PRINT "JE TROUVE, EN &HDFFF, LA VALEUR ";
   HEX$(PEEK(&HDFFF))
90 DATA 86,1A,C6,20,1E,89,B7,DF,FE,F7,DF,FF,
   39
```

Programme Basic d'implantation

On entre ainsi dans le mode étendu. Par exemple, les deux lignes :
STA $DFFE
STB $DFFF
stockent les valeurs contenues dans A et B respectivement aux adresses $DFFE et $DFFF. Le code-objet de ces instructions est :
B7 DFFE (B7 pour STA en mode étendu)
F7 DFFF (F7 pour STB en mode étendu)

Il suffit alors d'implanter et de lancer la routine réalisant ces opérations à l'aide du programme Basic d'implantation (ci-contre). Ce programme se termine par deux lignes demandant l'affichage des valeurs contenues aux adresses spécifiées : elles ont bien été échangées. La dernière instruction machine, 39, correspond à RTS (retour au système). C'est elle qui redonne la main au Basic.



Dans une boîte, l'adresse d'une autre boîte...

Dans le mode d'adressage étendu, l'opérande de 16 bits représente l'adresse où se trouve la valeur concernée par cette instruction, ou encore l'adresse où l'on voudra ranger le résultat de l'exécution de cette instruction. Ce mode peut également être utilisé avec LD. Par exemple, LDA $DFFF ira chercher la valeur contenue en DFFF et la chargera dans l'accumulateur A. Il est possible de détourner plus encore le chemin de l'adressage en utilisant l'adressage étendu indirect. Dans ce cas, l'opérande est l'adresse où se trouve l'adresse contenant la valeur à traiter ! Ainsi, dans l'exemple précédent, l'adresse $DFFE contenant $20 et l'adresse $DFFF contenant $1A, si l'on écrit : LDA [$DFFE] (les crochets symbolisent l'adressage étendu indirect), on ira chercher en $DFFE, le premier octet, soit $20, puis à l'adresse suivante $DFFF, le deuxième octet, soit 1A. Ces deux octets forment alors l'adresse $201A où l'on ira chercher la valeur à charger dans A.

Pour appliquer ce mode d'adressage, il suffit d'implanter la routine ci-contre (*Un exemple d'adressage étendu indirect*), selon le même principe que la précédente implantation. Ce n'est qu'un exemple, chacun peut trouver ses propres applications.

Un exemple d'adressage étendu indirect

Mnémoniques	Code-objet	
ORG $A000		
LDA #$20	86 20	on charge A avec $20
LDB #$1A	C6 1A	on charge B avec $1A
STA $DFFE	B7 DF FE	on stocke A en $DFFE
STB $DFFF	F7 DF FF	on stocke B en $DFFF
LDA [$DFFE]	9F DF FF	on charge A avec le contenu de l'octet dont l'adresse se trouve en $DFFE et $DFFF
STA $DFFF	B7 DF FF	on stocke cette valeur en $DFFF
RTS	39	
END		

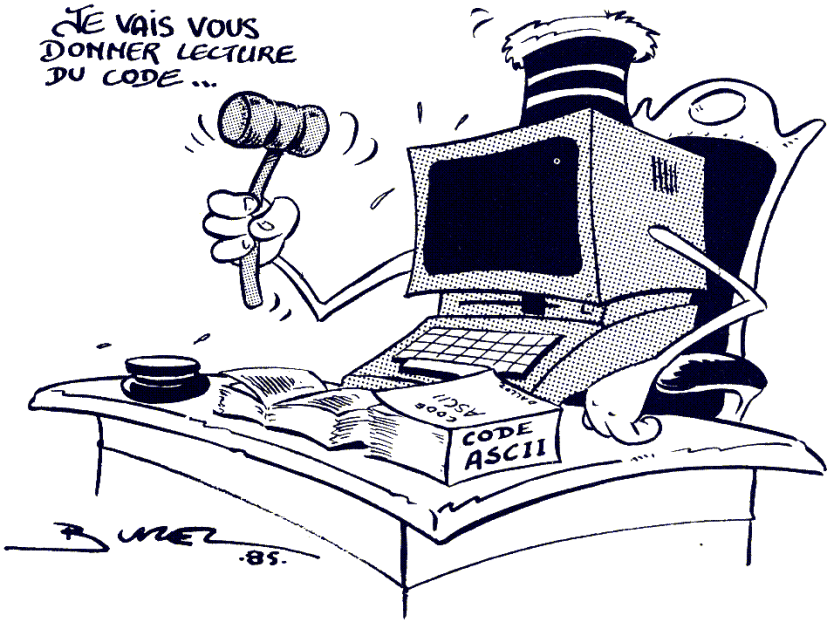
Jean-Paul CARRÉ

LECTURE D'UN CODE ASCII

POUR qu'un caractère frappé au clavier soit considéré comme un code ASCII, en Pascal, il faut une procédure. C'est l'objet de celle qui est proposée ici. Avec quatre modes d'entrée possibles, elle devient efficace et facile à utiliser.

Les codes ASCII jouent un rôle important dans les programmes, surtout si ces derniers sont un peu techniques. Par exemple, un correcteur (*patcher*, en anglais) qui modifie la valeur d'un octet en mémoire centrale ou sur disque, nécessite la saisie de valeurs ASCII, sous la forme la plus souple possible.

Un autre exemple est celui d'un programme correctement paramétré qui mémorise dans un fichier tous les caractères transmis à l'imprimante pour qu'elle passe en caractères expansés, normaux, réduits ou campactés. Là encore, la saisie des codes ASCII doit être facilitée. En effet, pour modifier un octet ou indiquer le code de contrôle d'une imprimante, il faut pouvoir entrer les valeurs sous plusieurs formes. Tel code est connu sous sa valeur décimale ou hexadécimale, tel autre l'est sous son symbole ASCII, ou sous son caractère de contrôle : autant de possibilités que doit proposer la fonction de saisie.



La procédure Pascal qui permet de saisir au clavier un caractère considéré comme un code ASCII est déclarée par :

procedure Lecture-ascii (*var* ascii : char ; *var* erreur : boolean) ;

Principaux caractères interceptés par les moniteurs des systèmes d'exploitation

Caractères	Code ASCII	Effet produit
ctrl @	0	Correspond au BREAK, et produit l'arrêt du programme en cours d'exécution.
ctrl C	3	Est associé à la fin de fichier, et termine prématurément la saisie au clavier.
ctrl F	6	Provoque le FLUSH, c'est-à-dire l'arrêt de l'affichage à l'écran.
ctrl H	8	Correspond au BACKSPACE, c'est-à-dire à l'effacement du dernier caractère entré.
ctrl M	13	Identique à la touche *return*
ctrl S	19	Provoque un STOP, c'est-à-dire la suspension de l'exécution du programme en cours.
ctrl X	24	Efface tous les caractères entrés sur la ligne en cours de saisie.

Le premier paramètre, ASCII, est un caractère qui est retourné par l'activation de cette procédure. Il contient le symbole entré au clavier, soit directement, soit par un de ses codes.

L'autre paramètre, ERREUR, est une variable logique, qui prend donc une des deux valeurs *false* (faux) ou *true* (vrai). La paramètre indique si l'entrée est correcte ou non. Dans le cas où ERREUR est positionné à *vrai*, la saisie ne correspond pas à un caractère, et la valeur de ASCII n'est pas modifiée par la procédure.

Contrôler les entrées

Afin que la saisie d'un symbole ASCII soit effectuée de manière simple, plusieurs modes d'entrée sont possibles. Le premier, le plus simple, consiste à entrer un caractère. Par ce mode, presque tous les symboles présents sur une touche du clavier utilisé peuvent être introduits. Ainsi, pour entrer le caractère 'S', il suffit de frapper la touche S suivie de *return*. Il en va de même pour les caractères de contrôle : le code *ctrl* G est entré, par exemple, par l'appui sur les touches *ctrl*, G puis *return*. Ce mode d'entrée présente toutefois deux limitations. D'une part, seuls les symboles disponibles au clavier peuvent être introduits (pour les autres caractères, il faut utiliser un autre mode). D'autre part, certains caractères de contrôle sont interprétés par le moniteur du système d'exploitation. La liste de ces caractères est variable d'un système à l'autre, mais ceux qui sont le plus souvent interceptés par les moniteurs sont réunis dans le tableau ci-dessus.

Avec le deuxième mode de saisie, tous les caractères de contrôle peuvent être entrés, mais sans être frappés directement au clavier. Ils doivent être précédés du symbole '^'. Ainsi, l'entrée de *ctrl* S se fait par '^S' suivi de la touche *return*.

Le troisième mode d'entrée des caractères ASCII s'effectue directement avec le code numérique. De cette façon, même les nombres compris entre 0 et 255 sont acceptés. Par exemple, le caractère de contrôle *ctrl* M est saisi en entrant 13. Les symboles dont le code numérique est compris entre 0 et 9 doivent être entrés sur deux chiffres, 00 à 09. L'oubli de cette règle entraîne une erreur : le chiffre entré est alors compris par la procédure comme le caractère correspondant à ce chiffre, et non comme le caractère dont ce chiffre représente le code. Ainsi, le caractère '0' qui a pour mode ASCII 48, est différent du caractère *ctrl*@dont le code ASCII est 0.

Enfin, le quatrième mode d'entrée est réservé aux caractères de contrôle. Il permet de les saisir sous la forme de leur mnémonique. Par exemple, à *ctrl* H qui est associé à la touche marquée *BackSpace* (ou flèche à gauche) correspond le mnémonique BS. La saisie a lieu indifféremment en majuscules ou en minuscules. La liste des mnémoniques peut être reconstituée à partir du texte de la procédure.

De nombreux aspects du langage Pascal apparaissent dans cette procédure. Ainsi, dès son entrée, un chaîne de caractères est lue au clavier et analysée par la procédure CATEGORIE-SAISIE. La valeur retournée est alors :

Lecture d'un code ASCII
Procédure Pascal
Auteur Thierry Chamoret
Copyright *LIST* et l'auteur

```
procedure lecture__ascii (var ascii : char ; var erreur : boolean) ;
type catesai = (caractere, controle, nombre, mnemonique, inconnu) ;
var entree : string ;
    code : integer ;
    function categorie__saisie (chaine : string) : catesai ;
    begin
      if lenght (chaine) < = 0
        then categorie__saisie : = inconnu
        else
          if lenght (chaine) = 1
            then categorie__saisie : = caractere
            else
              if chaine [1] = '^'
                then categorie__saisie : = controle
                else
                  if chaine [1] in ['0'..'9']
                    then categorie__saisie : = nombre
                    else categorie__saisie : = mnemonique
    end ;
    function valeur__decimale (chaine : string) : integer ;
    var valeur, i : integer ;
    begin
      valeur : = 0 ;
      for i : = 1 to length (chaine) do
        if chaine [i] in ['0'..'9']
          then
            valeur : = 10*valeur + ord (chaine [i]) − ord ('0') ;
      valeur__decimale : = valeur
    end ;
```

▶▶▶

```
function valeur__ascii (mnemonique : string) : integer ;
var valeur : integer ;
  procedure majuscules (var chaine : string) ;
  var i : integer ;
  begin
    for i : = 1 to length (chaine) do
      if chaine [i] in ['a'..'z']
        then
          chaine [i] : = chr (ord (chaine [i]) - ord ('a') + ord ('A'))
  end ;
  function val__0__a__15 (mnemonique : string) : integer ;
  begin
    val__0__a__15 : = - 1 ;
    if mnemonique = 'NUL'        then val__0__a__15 : =  0 else
    if mnemonique = 'SOH'        then val__0__a__15 : =  1 else
    if mnemonique = 'STX'        then val__0__a__15 : =  2 else
    if mnemonique = 'ETX'        then val__0__a__15 : =  3 else
    if mnemonique = 'EOT'        then val__0__a__15 : =  4 else
    if mnemonique = 'ENQ'        then val__0__a__15 : =  5 else
    if mnemonique = 'ACK'        then val__0__a__15 : =  6 else
    if mnemonique = 'BEL'        then val__0__a__15 : =  7 else
    if mnemonique = 'BS'         then val__0__a__15 : =  8 else
    if mnemonique = 'HT'         then val__0__a__15 : =  9 else
    if mnemonique = 'LF'         then val__0__a__15 : = 10 else
    if mnemonique = 'VT'         then val__0__a__15 : = 11 else
    if mnemonique = 'FF'         then val__0__a__15 : = 12 else
    if mnemonique = 'CR'         then val__0__a__15 : = 13 else
    if mnemonique = 'SO'         then val__0__a__15 : = 14 else
    if mnemonique = 'SI'         then val__0__a__15 : = 15
  end ;
  function val__16__a__127 (mnemonique : string) : integer ;
  begin
    val__16__a__127 : = - 1
    if mnemonique = 'DLE'        then val__16__a__127 : =  16 else
    if mnemonique = 'DC1'        then val__16__a__127 : =  17 else
    if mnemonique = 'DC2'        then val__16__a__127 : =  18 else
    if mnemonique = 'DC3'        then val__16__a__127 : =  19 else
    if mnemonique = 'DC4'        then val__16__a__127 : =  20 else
    if mnemonique = 'NAK'        then val__16__a__127 : =  21 else
    if mnemonique = 'SYN'        then val__16__a__127 : =  22 else
    if mnemonique = 'ETB'        then val__16__a__127 : =  23 else
    if mnemonique = 'CAN'        then val__16__a__127 : =  24 else
    if mnemonique = 'EM'         then val__16__a__127 : =  25 else
    if mnemonique = 'SUB'        then val__16__a__127 : =  26 else
    if mnemonique = 'ESC'        then val__16__a__127 : =  27 else
    if mnemonique = 'FS'         then val__16__a__127 : =  28 else
    if mnemonique = 'GS'         then val__16__a__127 : =  29 else
    if mnemonique = 'RS'         then val__16__a__127 : =  30 else
    if mnemonique = 'US'         then val__16__a__127 : =  31 else
    if mnemonique = 'DEL'        then val__16__a__127 : = 127
  end ;
begin
  majuscules (mnemonique) ;
  valeur : = val__0__a__15 (mnemonique) ;
  if valeur < 0 then valeur : = val__16__a__127 (mnemonique) ;
  valeur__ascii : = valeur
end ;
begin
  code : = -1 ;
  readln (entree) ;
  case categorie__saisie (entree) of
    caractere : if length (entree) = 1 then code : = ord (entree [1]) ;
    controle : if (length (entree) = 2) and (entree [2] in ['@'..'__'])
                 then code : = ord (entree [2]) - ord ('@') ;
    nombre : code : = valeur__decimale (entree) ;
    mnemonique : code : = valeur__ascii (entree) ;
    inconnu : ;
  end ;
  erreur : = not (code in [0..255]) ;
  if not erreur then ascii : = chr (code)
end ;
```

- soit *caractere*, si un symbole ASCII simple a été entré ;
- soit *controle*, si la chaîne saisie commence par le symbole '^' qui annonce un caractère de contrôle ;
- soit *nombre*, si la chaîne contient une valeur numérique correspondant à un code ASCII ;
- soit *mnemonique*, si l'entrée correspond au mnémonique d'un code ASCII ;
- soit, enfin, *inconnu*, si rien n'a été entré.

Trois fonctions essentielles

A partir de cela, la procédure principale, LECTURE-ASCII, oriente le traitement sur l'une des trois fonctions chargées de rechercher le code correspondant à la chaîne entrée.

La fonction VALEUR-DECIMALE est une version simplifiée de la procédure CONVENTIER déjà rencontrée dans *LIST* (voir *LIST* 4, page 49). Elle calcule la valeur numérique présente dans la chaîne passée en paramètre. La fonction VALEUR-ASCII recherche le code correspondant au mnémonique transmis en paramètre. Tout d'abord, elle convertit la chaîne en majuscules, la procédure MAJUSCULES effectuant ce traitement (MAJUSCULES est une version simplifiée de la procédure présentée dans *LIST* 3, page 61). Ensuite, une série de tests permet de déterminer la valeur du code associé au mnémonique. Il faut noter que cette série de comparaisons est partagée en deux parties, car de nombreux compilateurs n'admettent pas autant de comparaisons de chaînes dans une seule procédure.

Un fois que le traitement correspondant aux différents modes de saisie a été réalisé, la procédure principale LECTURE-ASCII vérifie la validité du code. Cela permet de mettre à jour la variable logique ERREUR, et, dans le cas où tout s'est bien passé, de mettre à jour le caractère ASCII afin qu'il soit retourné au programme appelant.

Avec ses quatre modes de saisie, cette procédure de lecture d'un symbole ASCII est assez souple. Elle peut être améliorée, par exemple, en prévoyant l'entrée des codes numériques dans une base autre que la base décimale. Il faut alors songer à faire suivre le nombre de l'initiale du nom de la base, soit B pour le binaire, O pour l'octal, ou H pour l'hexadécimal.

Thierry CHAMORET

PASSEZ-MOI L'EXPRESSION

AUTREFOIS, on les appelait des calculateurs, et malgré les apparences, les ordinateurs sont restés avant tout des machines à calculer. Voici un programme dont le but est d'évaluer les expressions. Vous me direz que votre Basic ou votre Pascal comprend déjà un programme de ce type. Oui, mais comment fonctionne-t-il ? Quels en sont les principes ?

En écrivant ce programme qui calcule n'importe quelle expression composée de parenthèses et des quatre opérations, j'avais deux objectifs. Tout d'abord, bien évidemment, cet utilitaire dispensera son utilisateur d'effectuer, à la main ou sur une calculatrice, des calculs plutôt pénibles. Je pense principalement à la simplification des résultats.

Mais ce programme est avant tout l'occasion d'exposer ici, sous une forme parlante, le fonctionnement du sous-programme le plus important de tout Basic : l'analyseur de formules. C'est cet analyseur qui retourne la valeur de toute expression mathématique, aussi compliquée soit-elle.

En ce qui nous concerne, comme nous l'avons dit, l'analyseur prendra en compte les parenthèses et les quatre opérations ; il est donc relativement simple, et par conséquent facile à décortiquer. En raison de cette simplicité, nous nous contenterons (modestement) de l'appeler *évaluateur* dans la suite de l'article.

Première approche : examinons dans ses grandes lignes le mode de fonctionnement du programme. Au début, l'expression entrée au clavier à la ligne 50 est légèrement modifiée de façon à ce que le traitement qu'elle va subir soit facilité. C'est ainsi qu'on lui rajoute un indicateur de fin, le signe du dièse, avec, à la ligne 110, l'affectation A$ = A$ + ''#''. Pour des raisons du même ordre, les espaces qu'elle contient éventuellement sont éliminés grâce au sous-programme de la ligne 1310 (*).

La nouvelle expression obtenue, toujours notée A$, est décomposée par caractère grâce à un pointeur H ; le caractère sélectionné est placé dans la variable C$, comme on le voit aux lignes 610, 720 et 980 avec la formule C$ = MID$(A$,H,1).

Quand l'expression a été évaluée, le résultat est exprimé sous la forme d'une fraction X/Y, et la variable C$ contient toujours le premier caractère d'A$ non encore utilisé. La ligne 130 s'explique alors aisément : lorsque toute l'expression a été évaluée, C$ doit contenir l'indicateur de fin, c'est-à-dire le signe « # ». Si ce n'est pas le cas, c'est que l'on se trouve en présence d'une erreur de syntaxe. Cela se produit par exemple si l'on entre l'expression : (1+2)(3+4). Dans ce cas en effet, le retour se fait sur la seconde parenthèse ouvrante à cause de l'absence de signe entre les deux facteurs.

(*) C'est la seule partie du programme comprenant — par souci d'esthétique ! — des instructions vraiment particulières à mon Basic. Voilà pourquoi j'ai donné une autre version des lignes 1320 à 1360 n'utilisant que des instructions très standard.

Un peu de respect pour les priorités

L'architecture de l'évaluateur, que nous allons maintenant décrire, reflète fidèlement la notion de priorité des opérateurs arithmétiques. Cette notion très importante est décrite dans pratiquement toutes les notices d'ordinateurs ou de calculatrices. Pour les expressions autorisées ici, les priorités, classées par ordre décroissant, sont les suivantes :

- ouverture de parenthèses et nombres
- changement de signe
- multiplication et division
- addition et soustraction.

Cette hiérarchie est en fait très importante puisqu'elle permet d'attribuer à toute expression une valeur et une seule. Soit, par exemple, à calculer 2*5+7/8. Dans un premier temps, l'évaluateur va calculer 2*5 et 7/8 (priorité absolue) et ce n'est qu'ensuite qu'il fera la somme des deux résultats obtenus. Si l'on attribuait à l'opérateur de l'addition une priorité supérieure à celui de la multiplication, l'expression serait calculée

Évaluateur de formules
Programme Basic
Auteur Bernard Kokanosky
Copyright *LIST* et l'auteur

```
10 PROC"Init"
20 REPEAT
30   PRINT STRING$(39,"~")
40   ER=0:P=-1:H=1
50   INPUT"▦Expression:";A$
60   PROC"Evalue"
70   PROC"Resultat"
80 UNTIL 0
90 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
100 DEF PROC"Evalue"
110 A$=A$+"#"
120 PROC"Espace":PROC"Niveau + -"
130 IF C$<>"#" ER=1:PROC"Err":ENDIF
140 ENDPROC
150 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
160 DEF PROC"Resultat"
170 PRINT:PRINT "Valeur:";X;
180 IF Y<>1 PRINT "/";Y;:ENDIF
190 PRINT:X0=X:Y0=Y
200 ENDPROC
210 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
220 DEF PROC"Niveau + -"
230 PROC"Niveau * /"
240 WHILE C$="+" OR C$="-" DO
250   J=1-(C$="-"):PROC"Sauve"
260   H=H+1:PROC"Niveau * /"
270   PROC"Recupere"
280   IF J=1 PROC"+" ELSE PROC"-":ENDIF
290 WEND
300 ENDPROC
310 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
320 DEF PROC"Sauve"
330 P=P+1:S(P)=J:P(P)=X:Q(P)=Y
340 ENDPROC
350 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
360 DEF PROC"Recupere"
370 J=S(P):A=P(P):B=Q(P):P=P-1
380 ENDPROC
390 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
400 DEF PROC"Niveau * /"
410 PROC"Signe"
420 WHILE C$="*" OR C$="/" DO
430   J=3-(C$="/"):PROC"Sauve"
440   H=H+1:PROC"Signe"
450   PROC"Recupere"
460   IF J=3 PROC"*" ELSE PROC"/":ENDIF
470 WEND
480 ENDPROC
490 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
500 DEF PROC"Signe"
510 PROC"Saute +"
520 IF C$="-" DO
530   H=H+1:PROC"Signe":X=-X
540 ELSE PROC"Facteur"
550 ENDIF
560 ENDPROC
570 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
580 DEF PROC"Saute +"
590 H=H-1
600 REPEAT
610   H=H+1:C$=MID$(A$,H,1)
620 UNTIL C$<>"+"
630 ENDPROC
640 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
650 DEF PROC"Facteur"
660 IF C$="(" DO
670   H=H+1:PROC"Niveau + -"
680   IF C$<>")" ER=2:PROC"Err":ENDIF
690 ELSIF C$="$" X=X0:Y=Y0
700 ELSE PROC"Nombre"
710 ENDIF
720 H=H+1:C$=MID$(A$,H,1)
730 ENDPROC
740 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
750 DEF PROC"Init"
760 CLS:CLR:MODE 1:X0=0:Y0=1
770 DIM S(20),P(20),Q(20),ER$(4)
780 FOR J=0 TO 4:READ ER$(J):NEXT J
790 ENDPROC
800 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
810 DATA Erreur Basic
820 DATA Erreur de syntaxe
830 DATA Pas de ')' en sous expression
840 DATA Division par zero
850 DATA Pas de nombre
860 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
870 DEF PROC"Err"
880 BEEP 5:PRINT
890 PRINT ER$(ER):PRINT:PRINT A$
900 PRINT TAB(H-1)"↑"
910 PRINT "Appuyez sur une touche:";
920 INKEY X$:PRINT:RUN 20
930 ENDPROC
940 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
950 DEF PROC"Nombre"
960 H0=H:H=H-1:K=-1
970 REPEAT
980   H=H+1:K=K+1:C$=MID$(A$,H,1)
990 UNTIL C$<"0" OR C$>"9"
1000 N$=MID$(A$,H0,K)
1010 IF N$="" ER=4:PROC"Err":ENDIF
1020 X=VAL(N$):Y=1:H=H-1
1030 ENDPROC
1040 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1050 DEF PROC"+"
1060 U=A*Y+B*X:V=B*Y:PROC"Pgcd"
1070 ENDPROC
1080 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1090 DEF PROC"-"
1100 U=A*Y-B*X:V=B*Y:PROC"Pgcd"
1110 ENDPROC
1120 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1130 DEF PROC"*"
1140 U=A*X:V=B*Y:PROC"Pgcd"
1150 ENDPROC
1160 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1170 DEF PROC"/"
1180 U=A*Y:V=B*X:PROC"Pgcd"
1190 ENDPROC
1200 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1210 DEF PROC"Pgcd"
1220 IF V=0 ER=3:PROC"Err":ENDIF
1230 J=U:K=V
1240 REPEAT
1250   C=V:V=U MODULO V:U=C
1260 UNTIL V=0
1270 X=J/C:Y=K/C
1280 IF Y<0 X=-X:Y=-Y:ENDIF
```

```
1290 ENDPROC
1300 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1310 DEF PROC"Espace"
1320 J=INSTR(A$," ",1)
1330 WHILE J>0 DO
1340   A$=IN$(A$,J-1,J+1,"")
1350   J=INSTR(A$," ",1)
1360 WEND
1370 ENDPROC
1380 '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

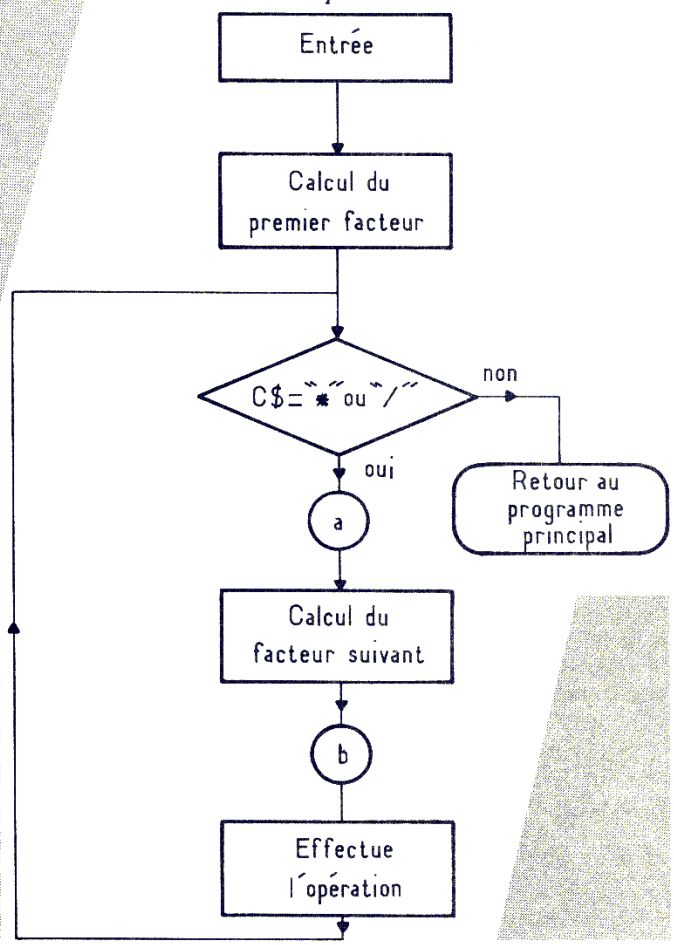
Les lignes 1320 à 1360 réécrites dans un Basic plus standard :

```
1320 L=LEN(A$)
1330 FOR J=1 TO L
1340   C$=MID$(A$,J,1)
1350   IF C$=" " A$=LEFT$(A$,J-1)+RIGHT$
(A$,L-J):L=L-1:J=J-1
1360 NEXT J
```

Quelques exemples d'exécution

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Expression:12/(3-56*1/-7)

Valeur: 12/ 11
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Expression:(1+1/2)(3/4+9/7)

Erreur de syntaxe

(1+1/2)(3/4+9/7)#
        ↑
Appuyez sur une touche:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Traitement des produits et des quotients



tout autrement, comme 2*(5+7)/8, résultat 3 au lieu de 10.875.

En résumé, on voit que, lors du calcul d'une expression, les nombres et les sous-expressions (ce qui se trouve entre parenthèses) sont évalués en premier ; les changements de signe sont pris en compte avant les produits et les quotients, et ce n'est qu'en dernier que sont calculés les sommes et les différences.

A noter que * et / ont une même priorité, ce qui signifie qu'ils seront traités selon l'ordre dans lequel ils apparaissent dans l'expression à évaluer.

La construction de tout évaluateur se fait en partant du niveau de priorité le plus élevé et en allant vers les niveaux les plus faibles. Dans le cas qui nous occupe, le traitement du niveau le plus élevé sera effectué par la procédure « Facteur » qui traite le cas des nombres (ligne 700) et celui des sous-expressions (ligne 670). On a incorporé au programme un dispositif qui en facilite l'utilisation : écrire $ revient à reprendre le dernier résultat (voir ligne 690).

On remarquera par ailleurs que pour calculer une sous-expression, il suffit de sauter la parenthèse ouvrante (H = H + 1 à la ligne 670) et de calculer ce qui vient ensuite grâce à l'évaluateur (procédure « Niveau + − »). Autrement dit, notre évaluateur est récursif : il s'appelle lui-même. Le retour se fait alors avec une parenthèse fermante dans C$ (d'où la ligne 680).

Plus ou moins ou ni plus ni moins

En descendant l'échelle des niveaux, nous rencontrons le problème du signe du facteur. Si ce signe est positif, on l'enjambe tout simplement (ligne 510). Si ce n'est ni le signe *plus* ni le signe *moins*, on exécute la procédure « Facteur » (ligne 540). Enfin, dans le cas du signe *moins*, il suffira de le sauter, de réutiliser la procédure « Signe » car on peut avoir écrit ensuite un signe + ou −, et de changer enfin le signe du résultat (ligne 530). On remarque, encore, l'utilisation de sous-programmes récursifs.

Arrivé à ce stade, nous devons faire calculer les produits et les quotients. Comme les opérations arithmétiques correspondantes ont toujours deux arguments, l'organigramme est très simple à mettre au point (figure ci-dessus).

Le calcul des divers facteurs doit se faire en traitant tout ce qui possède un niveau de priorité supérieur ou égal à ceux de la multiplication et de la division. Ce calcul passe par la procédure « Signe ». La programmation de « Niveau * / » est alors relativement simple. On remarque dans ce sous-programme le calcul de J qui vaudra 3 si l'opération est une multiplication, et 4 s'il s'agit d'une division. En effet, dans le Basic utilisé, la variable booléenne (C$ = "/") vaut −1 (vrai) ou 0 (faux) selon que le contenu de C$ est ou non le caractère de la division ("/").

Quiconque s'est déjà plongé dans la notice d'un ordinateur de poche ou

d'une calculatrice (surtout d'une Hewlett-Packard) sait qu'un évaluateur ne peut fonctionner qu'avec deux piles, l'une pour les nombres déjà calculés, l'autre pour les opérateurs rencontrés. Nous allons voir le pourquoi de leur utilisation après avoir donné les précisions suivantes :

- la pile des opérateurs est ici le tableau S(.) ;
- la pile des résultats partiels a été dédoublée, numérateur dans P(.) et dénominateur dans Q(.) ;
- le sommet des piles, c'est-à-dire le dernier niveau utilisé, est indiqué dans la variable P.

Dans l'organigramme précédent, nous avons remarqué que l'évaluateur devait effectuer le produit ou le quotient de deux termes, le premier représenté par la fraction X/Y (au point a), le second représenté par X/Y (au point b), le code de l'opération étant stocké dans la variable J (au point a). Il est alors évident que le calcul du second opérateur va éventuellement détruire J ainsi que le premier résultat, même si l'on prend la précaution de le stocker dans A/B et non pas dans X/Y. Nous allons voir pourquoi en prenant un exemple.

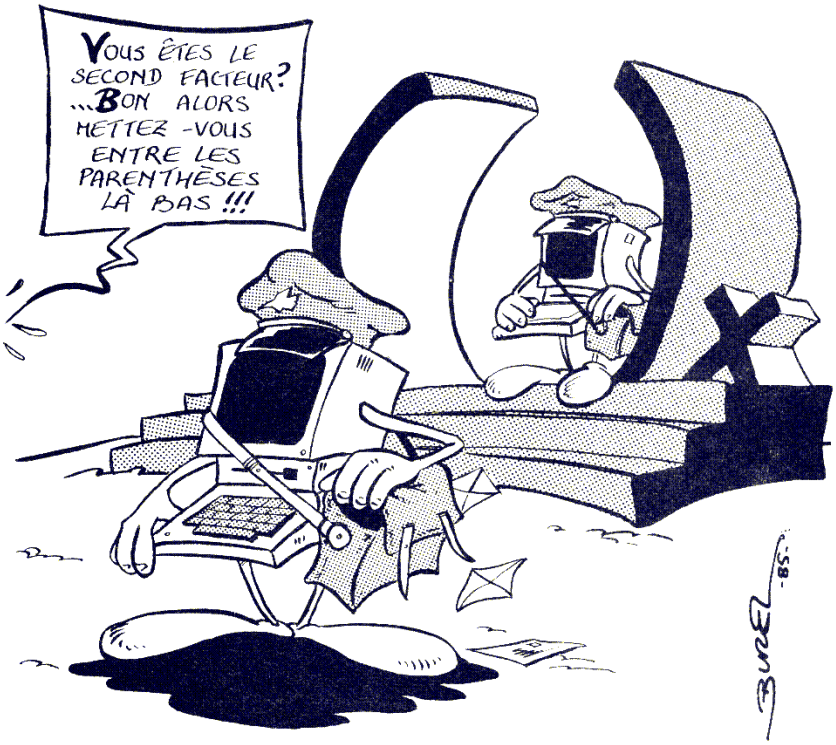
Soit à calculer 2*(3/4). Notre évaluateur va calculer le premier facteur 2, d'où un résultat A/B avec A = 2 et B = 1 , le code opératoire étant celui de la multiplication, autrement dit J = 3.

Comment préserver les anciennes valeurs ?

Pour le calcul du second facteur, les lignes 660 et 670 vont nous renvoyer à « Niveau + − » qui nous fera de nouveau passer dans « Niveau * / ». Le premier terme sera calculé, ici 3 d'où A/B avec A = 3 et B = 1, le code opératoire étant celui de la division : J = 4. Les anciennes valeurs, pourtant indispensables, de A, B, J sont irrémédiablement perdues et le résultat obtenu sera complètement faux.

C'est donc l'utilisation récursive de l'évaluateur qui va nous imposer la sauvegarde du premier résultat et du code opératoire en pile (Procédure « Sauve ») et leur récupération (Procédure « Récupère ») avant d'effectuer l'opération. (La pile S est celle des opérateurs, la pile P celle des numérateurs et la pile Q celle des dénominateurs.)

Lors du calcul de 2*(3/4), les choses se passeront donc de la façon suivante.



— Calcul du premier facteur et sauvegarde en pile :

avant le calcul de 3/4

pile S.....3
pile P.....2
pile Q.....1
P = 0

— Calcul du premier facteur de la parenthèse et sauvegarde :

avant le calcul de 4

pile S.....3 4
pile P.....2 3
pile Q.....1 1
P = 1

— Le calcul de 4 est alors effectué : X = 4 et Y = 1, et retour ligne 450. Il y a récupération de ce qui se trouve au sommet de la pile : A = 3, B = 1 et J = 4. D'où un nouvel état des piles :

pile S.....3
pile P.....2
pile Q.....1
P = 0

L'opération est effectuée, d'où X = 3 et Y = 4 et retour par 480 à la ligne 450, suivant celle qui avait lancé le calcul du second facteur (3/4).

Au retour en 450, la pile est vidée et l'on retrouve A = 2, B = 1, J = 3, avec X = 3 et Y = 4. Le produit est effectué pour donner le résultat X = 3 et Y = 2. Cette fois, tout fonctionne correctement.

Il ne reste plus qu'à traiter le cas des opérations de moindre priorité + et −. Il n'y a pas grande différence avec ce qui se passe pour la multiplication et la division, aussi laisserai-je le soin au lecteur d'étudier le sous-programme « niveau + − » qui constitue bien le point d'entrée de notre évaluateur.

Quelques précisions avant de terminer. Il faut savoir que le programme peut accepter jusqu'à 20 imbrications de parenthèses (dimensionnement de la ligne 770), mais qu'il est tout à fait possible d'augmenter cette limite. Cependant, le lecteur utilisant un Basic classique qui remplacerait PROC par GOSUB et ENDPROC par RETURN devrait veiller au nombre maximum d'imbrications de GOSUB-RETURN dans son Basic. L'utilisation récursive de notre évaluateur peut poser quelques problèmes s'il y a trop de sous-expressions emboîtées. Si ON ERROR GOTO existe dans ce Basic, il sera possible d'écrire un ON ERROR GOTO 880 dans le sous-programme « Init », et la saturation de la pile des adresses de retour des sous-programmes sera signalée par l'erreur 0 : « Erreur Basic ».

Après ces quelques explications, j'espère que beaucoup commenceront à avoir une idée plus précise du fonctionnement interne des Basic (comme des calculatrices en notation algébrique) et pourront, pourquoi pas, envisager la construction d'analyseurs de formules plus perfectionnés que celui de leur Basic, travaillant par exemple avec des formules hyperboliques, ou manipulant des nombres de plusieurs centaines de chiffres.

Bernard KOKANOSKY

EULER ACCÉLÈRE LA CONVERGENCE

CERTAINES suites ne s'approchent de leur limite qu'après un temps très long. Pour accélérer la convergence, la méthode d'Euler est d'une incontestable efficacité. Elle l'est particulièrement lorsque la convergence a lieu par oscillation autour de la limite.

L'une des applications mathématiques les plus évidentes de l'ordinateur consiste à obtenir une valeur approchée de la limite l d'une suite convergente u_n en calculant simplement suffisamment de termes consécutifs jusqu'à obtenir une stabilisation des résultats.

Un coup au-dessus, un coup en dessous

Malheureusement il arrive que la suite étudiée soit assez capricieuse et n'« atteigne » sa limite (à la précision de la machine près) qu'au bout de temps prohibitifs. La méthode suivante, due à Leonhard Euler, permet assez souvent de ramener u_n à de meilleurs sentiments par une transformation assez simple, qui n'en altère évidemment pas la limite. Elle est particulièrement efficace lorsque u_n oscille sans cesse autour de l : un coup au-dessus, un coup en dessous et ainsi de suite. Sous des conditions assez larges, on montre alors que la transformée e_n de u_n va se précipiter vers l sans se permettre aucune circonvolution intempestive. La transformation d'Euler est simple à programmer. Nous allons en exposer le principe en l'appliquant à un cas exemplaire, celui du calcul de l'intégrale :

$$I = \int_{-00}^{+00} \frac{\sin x}{x}\, dx$$

dont la valeur est — bien entendu — le nombre π. Si les résultats fournis par une calculatrice munie d'une touche « Integrate », HP-15C par exemple, ne sont pas très bons, cela tient à l'intégrale elle-même, qui a un bien vilain comportement.

Pour la calculer, une idée fort naturelle consiste à utiliser l'une des méthodes classiques d'intégration qui passe par les intégrales partielles :

$$u_n = \int_{-n\pi}^{n\pi} \frac{\sin x}{x}\, dx$$

et à étudier la limite de u_n, qui est évidemment I. Seulement la suite u_n possède justement le caractère oscillant annoncé : pour n = 30, nous verrons par exemple que u_n saute encore de 3,12 à 3,16 ! Pour obtenir quatre décimales exactes, il y faudrait un sacré bout de temps...

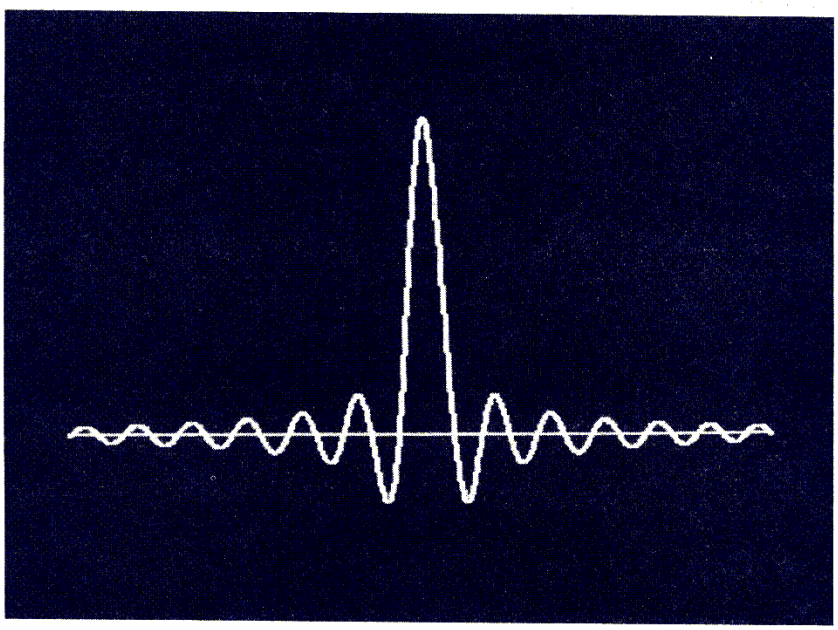
Heureusement voici Euler qui va tout régulariser, tout lisser et donnera la bonne valeur en un temps très raisonnable. L'idée fondamentale est la suivante : à partir de la suite (u_0, u_1, u_2, etc.), on définit une première suite (v_0, v_1, v_2, etc.) par un certain procédé (P), puis on réitère (théoriquement) ce procédé une infinité de fois, d'où l'on déduit la suite d'Euler e_n qui convergera vers I bien plus vite que u_n.

L'algorithme de (P) est simple ; on considère d'abord les différences successives d_n entre les termes de u_n : $d_0 = u_0$, $d_1 = u_1 - u_0$, $d_2 = u_2 - u_1$, etc. Ces différences mesurent les écarts entre les différentes approximations de I que constituent les u_n. Dans le cas présent, ces écarts sont alternativement positifs et négatifs, grossièrement égaux en valeur absolue. Il est donc naturel d'essayer de les neutraliser en les sommant deux par deux, pour donner des nombres assez petits en valeur absolue. A savoir :

$q_0 = (d_0 + d_1)/2$, $q_1 = (d_1 + d_2)/2$,
$q_2 = (d_2 + d_3)/2$, etc.

puis on définit la suite v_n, transformée

EULER ACCÉLÈRE LA CONVERGENCE



Pour osciller, ça oscille

de u_n par (P), comme celle dont les q_n sont justement les écarts :

$v_0 = q_0$, $v_1 = v_0 + q_1$, $v_2 = v_1 + q_2$, etc.

On imagine bien volontiers que cette nouvelle suite v_n soit plus régulière que u_n. Admet-elle une limite ? Cette limite est-elle liée à l ?

Un calcul très simple fournit l'égalité :

$v_n = u_n + (d_{n+1} - u_0)/2$.

Il est alors clair que v_n admet une limite, à savoir $m = l - u_0/2$. Si l'on recommence la transformation (P) sur v_n, elle donnera une nouvelle suite w_n qui admettra comme limite $n = m - v_0/2 = l - (u_0 + v_0)/2$, etc.

On imagine la démarche générale : au bout d'un certain nombre d'applications du procédé (P), on obtiendra des suites formées de termes suffisamment petits pour être négligeables (pour la machine) — évidemment ceci est susceptible d'une démonstration mathématique rigoureuse, mais un peu délicate —, ce qui donne l'égalité approchée :

$l = (u_0 + v_0 + w_0 + \ldots)/2$.

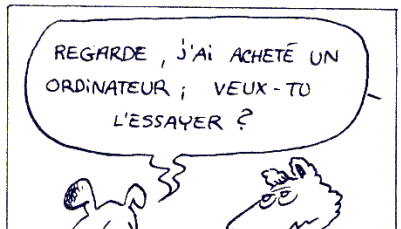
La suite d'Euler est donc simplement la suite définie par $e_0 = u_0/2$, $e_1 = (u_0 + v_0)/2$, $e_2 = (u_0 + v_0 + w_0)/2$, etc. Elle converge toujours vers la même limite que la suite u_n initiale, sans que l'on soit toujours assuré qu'il y ait réellement accélération de la convergence. Cela dépend des cas. Mais il en est très souvent ainsi lorsque u_n oscille avec régularité. Signalons encore que la suite d'Euler peut parfaitement converger même lorsque u_n diverge (voir ce qui se passe pour $u_{2m} = 1$, $u_{2m+1} = 0$: alors $d_n = (-1)^n$, $q_n = 0$, $v_n = 0$, $w_n = 0$, etc. et la suite e_n converge vers 1/2 ; une belle légende voudrait que l'ingénieur Eiffel ait utilisé une approximation de ce genre pendant la construction de sa Tour ; *se non è vero...*).

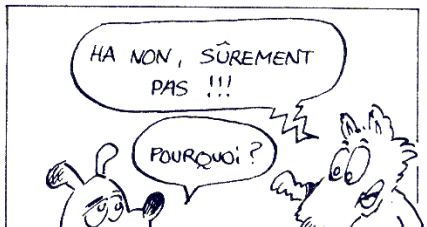
Un grand avantage de cette méthode est que l'on n'utilise ce dont on a besoin qu'au moment où l'on en a besoin. Il serait impossible (et stupide !) de rentrer en machine tous les u_n alors que, pour calculer par exemple e_{10}, on ne se servira que de $u_0, u_1, \ldots, u_{10}$. Le but recherché est d'afficher successivement côte à côte les u_n et les e_n à l'écran jusqu'à stabilisation des e_n. Dans la pratique, on n'a jamais besoin d'aller

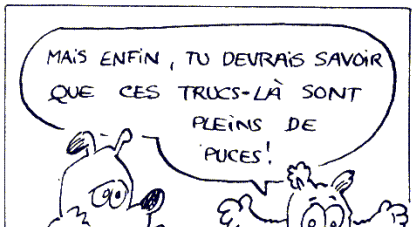
Pour les curieux... et les matheux

Il existe une formule donnant explicitement e_n en fonction des u_n. Elle utilise des coefficients binômiaux et peut servir pour la démonstration de la convergence des e_n vers l. La voici :

$$e_n = 2^{-n-1} \sum_{0}^{n} C_{n+1}^{k+1} u_k$$







```
10   CLS: DEFINT G,I,J,K,N
20   N=500: PI=3.1415927: A=0: H=PI/N
30   DIM E(N): S=0: E=0
40   FOR I=0 TO N
50       GOSUB 140: E(I)=U
60       PRINT I+1;: S=S+U: PRINT TAB(30) S;
70       IF I=0 GOTO 110
80       FOR J=I-1 TO 0 STEP -1
90           E(J)=(E(J)+E(J+1))/2
100      NEXT J
110      E=E+E(0)/2: PRINT TAB(50) E
120  NEXT I
130  PRINT: END
140  B=A+PI
150  X=A: GOSUB 250
160  U=Y: G=2
170  FOR K=1 TO N
180      G=6-G: X=X+H
190      GOSUB 250
200      U=U+G*Y
210  NEXT K
220  U=H*(U-Y)/3
230  A=B
240  RETURN
250  IF X<1E-2 THEN Y=1-X*X/6 ELSE Y=SIN(X)/X
260  Y=Y+Y
270  RETURN
```

Calcul de l'intégrale de sin(x)/x par la méthode d'accélération d'Euler

au-delà de 15 ou 20 ; l'exemple présenté ici va cependant jusqu'à u_{52} et e_{52}.

Une boucle après l'autre

Les nombres u_0, v_0, w_0,... occuperont successivement la mémoire nommée E(0), et les e_n seront stockés en E. Les différences successives d_n, données par les formules :

$$d_n = u_n - u_{n-1} = \int_{-n\pi}^{n\pi} \frac{\sin x}{x}\, dx$$

$$= \int_0^{n\pi} \frac{2 \sin x}{x}\, dx \quad (d_0 = 0)$$

seront entrées, chacune en son temps, dans les mémoires respectives E(i). Après l'introduction de d_n en E(n − 1), on remplacera le nombre d_{n-1} en mémoire E(n − 2) par le nombre $q_{n-1} = (d_{n-1} + d_n)/2$ et ainsi de suite. Au bout de cette boucle, le contenu de E(0) sera égal à $2(e_n - e_{n-1})$. Après affichage de e_n, une autre boucle commencera par l'appel du successeur u_{n+1}, etc.

Le calcul des u_n se fait par une routine élémentaire (ici la méthode de Simpson en 500 pas ; bien d'autres auraient pu faire l'affaire). Elle est située dans le sous-programme des lignes 140 à 240. L'algorithme d'Euler lui-même figure entre les lignes 30 et 130. Il peut évidemment être appliqué à bien d'autres choses qu'au calcul d'intégrales vicieuses. Sans doute n'est-il pas toujours aussi efficace que dans cet exemple. Mais reconnaissons que la sortie des résultats (voir ci-contre) est impressionnante : la valeur exacte 3.14159 est donnée dès le dix-septième pas, alors que u_n oscille encore entre 3.10 et 3.18 ; e_n reste stable jusqu'au bout (une variation de 10^{-5} entre les pas 19 et 42 est due, semble-t-il, à des problèmes d'arrondi), alors que u_{51} et u_{52} ne permettent même pas d'être sûr de la seconde décimale !

André WARUSFEL

Liste des cinquante-deux premières valeurs des suites u_n et e_n

n	u_n	e_n
1	3.70387	1.85194
2	2.83632	2.56102
3	3.34954	2.87126
4	2.98432	3.01349
5	3.26795	3.08024
6	3.03607	3.11199
7	3.23218	3.12724
8	3.06227	3.13461
9	3.21217	3.13818
10	3.07806	3.13993
11	3.19938	3.14078
12	3.08862	3.1412
13	3.19052	3.1414
14	3.09617	3.1415
15	3.18401	3.14155
16	3.10184	3.14158
17	3.17903	3.14159
18	3.10625	3.14159
19	3.17509	3.1416
20	3.10978	3.1416
21	3.17188	3.1416
22	3.11268	3.1416
23	3.16925	3.1416
24	3.11509	3.1416
25	3.16704	3.1416
26	3.11713	3.1416
27	3.16515	3.1416
28	3.11888	3.1416
29	3.16353	3.1416
30	3.12039	3.1416
31	3.16211	3.1416
32	3.12171	3.1416
33	3.16087	3.1416
34	3.12288	3.1416
35	3.15977	3.1416
36	3.12392	3.1416
37	3.15879	3.1416
38	3.12485	3.1416
39	3.15791	3.1416
40	3.12569	3.1416
41	3.15711	3.1416
42	3.12645	3.1416
43	3.15639	3.14159
44	3.12713	3.14159
45	3.15573	3.14159
46	3.12776	3.14159
47	3.15513	3.14159
48	3.12834	3.14159
49	3.15458	3.14159
50	3.12887	3.14159
51	3.15407	3.14159
52	3.12936	3.14159

GRAPHISMES A L'ENCRE

LE manuel de l'utilisateur de l'Amstrad ne dit pas tout sur les possibilités graphiques de la machine. Il est particulièrement succinct sur le mode d'encre graphique, mode pourtant simple à mettre en œuvre et qui permet des réalisations très colorées.

A chacun des trois modes d'écran de l'Amstrad correspond un nombre de colonnes de texte et des qualités de résolution du graphisme :

- le mode multicolore est en basse résolution, avec 20 colonnes de texte et 16 couleurs ;
- le mode normal est en résolution moyenne, avec 40 colonnes de texte et 4 couleurs ;
- le mode haute résolution dispose de 80 colonnes de texte et de 2 couleurs seulement.

Un autre mode, indépendant dans son principe de ces modes d'écran, propose des effets variés : le mode d'encre. Il n'est mentionné dans le guide de l'utilisateur qu'au chapitre 9, dans le paragraphe sur les « caractères de contrôle ». On y apprend, en fait, que le caractère dont le code ASCII est 23 permet de modifier le « Mode d'Encre Graphique », pour autant qu'il soit suivi d'un paramètre pouvant varier de 0 à 3 inclus. Le mode normal est actionné par 0, la valeur 1 place l'Amstrad en mode XOR, 2 en mode AND, 3 en mode OR. Ce que le manuel ne dit pas, c'est que ce paramètre doit également être manipulé comme un caractère ASCII. Ainsi, pour activer le mode d'encre graphique XOR en Basic, il faut écrire : PRINT CHR$(23);CHR$(1).

Le mode d'encre normal (20 colon-

Tableau 1
Correspondance entre les numéros de PEN et les couleurs

	Numéro de PEN	Couleur
0	0000	Bleu foncé
1	0001	Jaune vif
2	0010	Turquoise vif
3	0011	Rouge vif
4	0100	Blanc
5	0101	Noir
6	0110	Bleu vif
7	0111	Magenta vif
8	1000	Turquoise
9	1001	Jaune foncé
10	1010	Bleu pastel
11	1011	Rose
12	1100	Vert vif
13	1101	Vert pastel
14	1110	Bleu/jaune
15	1111	Rose/Bleu ciel

Tableau 2
Résultats des opérateurs AND, OR, XOR

Valeur du premier bit	Valeur du second bit	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Programme 1
Les modes d'encre de l'Amstrad

```
1 MODE 0
100 REM **** MODES D'ENCRAGE CPC ****
110 PRINT"0- NORMAL"
120 PRINT"1- XOR"
130 PRINT"2- AND"
140 PRINT"3- OR"
150 LOCATE 1,6:INPUT"MODE ";C$
160 C=VAL(C$):IF C<0 OR C>3 THEN 150
170 LOCATE 1,8:INPUT"PEN ";P
180 CLS:PRINT CHR$(23);CHR$(0)
190 '
200 REM *** HORIZONTAL ***
210 X0=150:Y0=150:X1=490
220 FOR Y=Y0 TO Y0+100 STEP 2
230 MOVE X0,Y:DRAW X1,Y,1
240 NEXT
250 REM *** VERTICAL ***
260 PRINT CHR$(23);CHR$(C)
270 X0=265:Y0=70:X1=385
280 FOR Y=Y0 TO Y0+250 STEP 2
290 MOVE X0,Y:DRAW X1,Y,P
300 NEXT
310 CALL &BB18:RUN
320 END
```

Explications du programme 1

Dans le cas du mode AND, le fond de l'écran a la couleur 0000 (PAPER 0). Une barre horizontale jaune qui correspond au PEN numéro 0001 traverse l'écran. Le choix du PEN numéro 1000 (turquoise) pour la barre verticale a pour résultat une barre verticale de couleur bleue partout (PEN 0000). En effet, sur la zone de fond, on a 0000 AND 1000, soit 0000 ; au croisement des deux barres colorées, on a 0001 AND 1000, soit 0000, là encore.

Dans le cas du mode XOR, le fond d'écran est en PEN 0000, la barre horizontale toujours en PEN 0001. Le choix du PEN numéro 1001 (jaune foncé) a pour effet de colorer la zone de fond en jaune, soit un PEN numéro 0001, car 0000 XOR 0001 a pour résultat 0001. En revanche, c'est la couleur correspondant à 0001 XOR 1001, soit le turquoise (1000) qui apparaît au croisement des barres.

Pour ce qui est du mode normal, il est le seul à ne pas faire intervenir de changement de couleur : la barre verticale recouvre normalement la barre horizontale en conservant partout la couleur choisie. Ce mode est donc équivalent à un 1111 AND numéro de PEN.

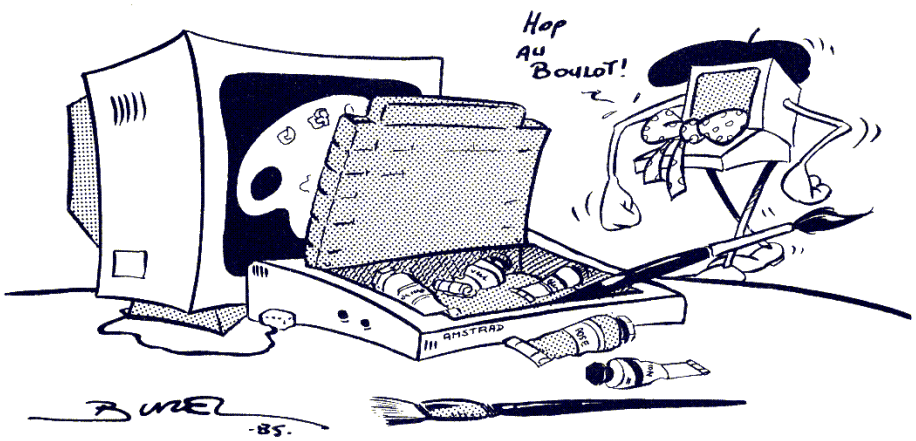
Enfin, la ligne 1 du programme peut être modifiée pour essayer d'autres modes d'écran.

nes texte), établi par CHR$(0), laisse le choix entre 16 couleurs simultanées, 16 PEN. Chacun d'eux est reconnu par un numéro, de 0 à 15 en décimal, soit de 0000 à 1111 sur quatre chiffres binaires (bits). Le tableau 1 donne les correspondances entre les numéros de PEN et les couleurs à la mise en marche de l'ordinateur.

Les autres modes interviennent sur les couleurs par le changement de valeur des bits de PEN : les bits sont comparés deux à deux pour donner une nouvelle valeur de PEN. La table 2 expose les résultats des opérations AND, XOR et OR appliquées aux valeurs binaires que prendront les bits.

Le programme 1 met en évidence les résultats de l'emploi du mode d'encre (voir les explications du programme 1). Quant au programme 2, il est un exemple d'effet graphique remarquable : la simulation du déplacement d'un élément graphique sur un autre sans altération du dessin. Il reste à chacun la possibilité de faire ses propres expériences.

Jean-Pierre LALEVÉE



Programme 2
Démonstration graphique

```
100 REM *** demo graphique ***
110 CLS
120 PRINT CHR$(23)CHR$(1):REM mode XOR
130 FOR x=120 TO 480 STEP 16
140 MOVE x,0:DRAW x,399,3
150 NEXT
160 FOR i=100 TO 500 STEP 2
170 FOR j=1 TO 2
180 MOVE i,200:DRAWR 12,0,2:MOVER -6,6:DRAWR 0,-12
190 NEXT:NEXT
```

LA RENCONTRE DE CHARLES ET ADA

UN jour de juin 1833, l'inventeur des machines à différences, Charles Babbage, rencontra la fille du poète Byron, Ada de Lovelace. Les relations — pas toujours paisibles — qui suivirent cette rencontre rendirent un grand service à la science et particulièrement à la programmation.

Vers 1833, l'astronome Charles Babbage, né le lendemain de Noël 1791, avait déjà fort bien utilisé la première moitié de ses quatre-vingts années de vie — somme toute assez malheureuse et pleine d'intrigues (1). Il avait déjà tenté de mettre au point le premier des deux projets grâce auxquels il mérite d'être considéré comme l'un des pères des ordinateurs modernes : les *Difference Engines* (machines à différences).

Elles avaient été conçues afin de dresser et d'imprimer automatiquement des tables de fonctions mathématiques (trigonométriques, exponentielles,...), mais non de pouvoir traiter des opérations arithmétiques à la demande comme avec une calculatrice de Pascal. Au cours des longs conflits qui opposèrent cet homme atrabilaire à ses employés et à son gouvernement (à qui il voulait faire supporter les frais inouïs de construction du premier modèle), Babbage conçut subitement vers 1833 sa *machine analytique*, d'une tout autre importance pour nous. Les premiers plans de ce qui restera comme son chef-d'œuvre datent, semble-t-il, de septembre 1834.

Déjà les cartes perforées

Lorsqu'ils prennent connaissance pour la première fois de la nature de la *machine analytique*, même les profanes sont surpris par l'idée simple et géniale. Pour réaliser ses projets, Babbage fut en effet amené à utiliser une technique très spectaculaire à base de cartes perforées. Cette technique fut essentiellement mise au point pour les métiers à tisser, entre 1801 et 1805, par Jacquard et Breton (d'après des idées anciennes de Vaucanson, vers 1750, elles-mêmes issues des recherches de deux soyeux lyonnais, Basile Bouchon en 1725 et Falcon trois années plus tard). Cet artifice

(1) La première moitié de la vie de Charles Babbage est racontée dans LIST *11, pages 19 à 22.*

fondamental joua le rôle moteur dans sa définition des plans d'un prototype de calculateur « universel », c'est-à-dire capable en principe d'exécuter automatiquement n'importe quel algorithme.

Pour prendre une image brutale peut-être et un peu trop approximative, on peut dire qu'il y a, entre les *Difference Engines* et l'*Analytical*, tout ce qui sépare un tapis roulant d'une voiture. Les deux mènent sans effort d'un point à un autre, mais l'un est restreint par nature à certains déplacements précis, alors que l'autre ne connaît guère de limites si on sait la diriger. Comment ne pas comprendre qu'il s'agit de deux niveaux complètement séparés du concept d'automatisation ?

Même si l'*Analytical Engine* ne fut construite, comme on l'a déjà vu pour les précédentes, que d'une façon extrêmement partielle (l'unité centrale de calcul fut complétée après la mort de Babbage, en 1880, par son fils Henry), même si elle ne servit que deux fois, le 21 janvier 1888 pour calculer et imprimer 44 multiples de *pi*, puis après quelques modifications en 1910 pour des tables astronomiques, même si son utilisation fut rendue presque impossible par des difficultés d'ordre technologique insurmontables pour l'époque, on peut considérer que le principe de la *machine analytique* était tout à fait viable. En un sens, le Mark I, la calculatrice « Harvard-IBM » qui fit sensation en 1944, n'en fut simplement qu'une version plus performante, sans doute parce qu'elle utilisait la puissance électrique (Babbage n'avait pensé qu'à la vapeur).

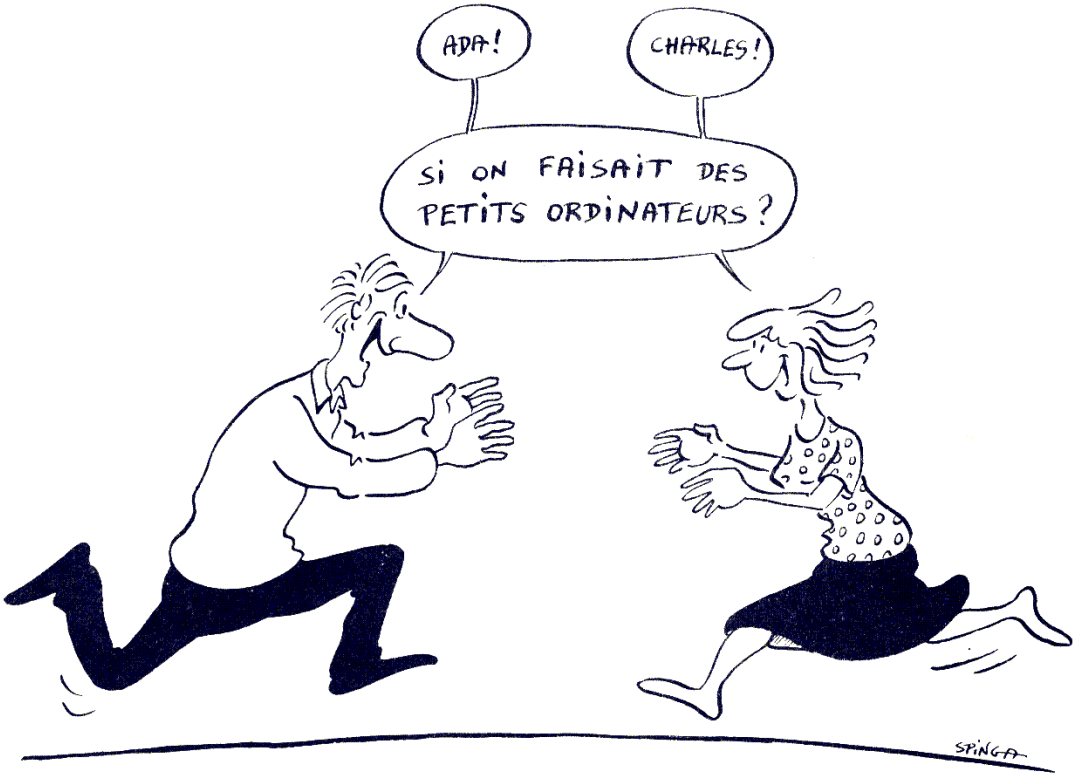
50 000 chiffres en mémoire !

Déjà, l'*Analytical Engine* avait tout ce qui fait l'essentiel d'un calculateur universel : des périphériques d'entrée et de sortie dont une imprimante, une unité centrale de calcul (*mill*, le moulin), une unité de commande (*control unit*), et surtout une mémoire gigantesque (*store*, littéralement le magasin) de mille compteurs décimaux à cinquante chiffres. Elle devait additionner et soustraire en une seconde, multiplier et diviser en une minute. Nous ne pouvons décrire ici en détail les composants du calculateur faute de place. Mais au simple énoncé de ces termes évocateurs, on voit que cette machine était puissante et avait déjà été conçue selon la structure fondamentale que l'on retrouve sans exception dans un ordinateur d'aujourd'hui. Entre autres aptitudes remarquables, elle était capable d'effectuer des boucles (en Basic, on dirait : FOR I = 1 TO N... NEXT I).

Son concepteur en était tout à fait conscient : nous en avons la preuve dans les textes qu'il lui consacra, et surtout dans le plus précieux de tous les documents inspirés par sa machine : la traduction, augmentée de notes capitales, par Ada de Lovelace, qui parut dans les « Taylor's Scientific Memoirs » de 1843, à partir d'un long article de Luigi Federico Menabrea dans la livraison d'octobre 1842 de la Bibliothèque universelle de Genève. (Des indications biographiques sur les auteurs des articles de 1842 et 1843 sont données en encadré, page 45.) Les notes, tout particulièrement, sont considérées aujourd'hui comme la première publication dédiée à l'idée alors toute neuve (et pour longtemps !) de programmation automatique.

C'est en effet une contribution sans prix qu'apporte à Babbage la célèbre comtesse Augusta Ada de Lovelace (1815-1852), née Byron, qui consacra les dix dernières années de sa vie à mettre au clair des exemples de suites des mouvements de cartes destinés à obtenir tel ou tel calcul. Elle inventa à ce propos (ou se contenta-t-elle d'expliciter d'après des suggestions de Babbage lui-même ? on l'ignore) la notion aujourd'hui fort banale de sous-programme qui, comme les itérations, permet d'utiliser autant de fois que nécessaire un même jeu d'instructions sans avoir à en prévoir plusieurs « copies » dans l'écriture de l'algorithme général.

Bien entendu, le seul fragment maté-



Les nombres de Jacques Bernoulli

Il y a environ 300 ans que le savant suisse Jakob (ou Jacques) Bernoulli (1654-1705), premier d'une nombreuse famille de mathématiciens, à l'origine de la *Combinatoire* et de nombreux résultats d'analyse et de géométrie, découvrit une suite remarquable de nombres qui portent son nom. Ils apparurent publiquement pour la première fois dans la deuxième partie de l'édition posthume de son grand livre sur l'art de conjecturer *(Ars Conjectandi,* 1713). Son compatriote Leonhard Euler en établit de nombreuses propriétés vers 1740. La Comtesse Ada de Lovelace, dans un mémoire de 1843 sur la Machine de Babbage, les utilisa comme cobayes pour prouver que l'*Analytical Engine* était capable de résoudre des problèmes complexes de calcul. (Babbage a peut-être été lui-même à l'origine de ce choix.)

Dans l'annexe G qui constitue l'une de ses contributions à cet article, elle commence par rappeler la définition usuelle des nombres de Bernoulli : la fonction définie par $f(0) = 1$ et, pour x non nul, par $f(x) = x/(e^x - 1)$, possède ce que l'on appelle techniquement un développement en série entière autour de 0 — outil essentiel, par exemple, des recherches de Newton — aux curieuses propriétés. Si l'on note ce développement sous la forme usuelle (valide pour x inférieur en valeur absolue à 2π) :

$$f(x) = \sum_{n=0}^{+\infty} b(n)x^n/n\,!,$$

on constate que les nombres b(n), naturellement égaux aux dérivées successives de f en 0, sont nuls pour n impair supérieur ou égal à 3, et satisfont à une étrange relation où interviennent les fameux coefficients du binôme, C(n, p) (étudiés notamment par Pascal, également l'un des pères de l'informatique). Sachant que $b(0) = 1$, pour tout entier m au moins égal à 1, on a l'égalité :

$$\sum_{p=0}^{p=m} C(m+1,p)\, b(p) = 0.$$

On donne ci-dessous une table des premières valeurs des nombres de Bernoulli, b(p), que vous pouvez obtenir très facilement par un programme Basic.

Table des premières valeurs des nombres de Bernoulli

p	0	1	2	3	4	5	6	7	8	9	10	11	12	13
b(p)	1	− 1/2	1/6	0	− 1/30	0	1/42	0	− 1/30	0	5/66	0	− 691/2730	0

La comtesse Ada prend d'abord soin de démontrer la relation en question, ou plus exactement une variante destinée à ne calculer que les b d'indice pair, les seuls qui ne soient pas nuls (en dehors du premier qui fait exception). Elle utilise, en fait, la notation B(2n − 1) pour représenter le nombre que nous écrivons aujourd'hui b(2n) ou B(n), mais il n'y a jamais eu d'accord complet entre les mathématiciens sur ce point. Ces nombres $B(1) = 1/6$, $B(3) = -1/30$, $B(5) = 1/42$, etc. sont liés par une égalité remarquable qu'elle note sous la forme : $0 = A(0) + A(1)B(1) + A(3)B(3) + \ldots + A(2n-1)B(2n-1)$ où les coefficients A(2p − 1) ne dépendent que de n et de p. Ils se prêtent facilement à un calcul récurrent.

Quelques-unes de leurs valeurs et des relations qui en permettent l'obtention se trouvent dans l'encadré ci-contre.

Il est donc facile de déterminer les uns après les autres les coefficients A(2p − 1). Si l'on connaît déjà les nombres de Bernoulli B(1), B(3), ..., B(2n − 3), la formule de la comtesse donne aussitôt B(2n − 1).

On rencontre partout, en mathématiques, les nombres b(p). Leur champ de prédilection est certainement la *combinatoire,* mais aussi l'analyse, notamment à cause d'une formule célèbre due à Euler et Mac-Laurin, pour calculer certaines sommes (on en utilise d'ailleurs certains cas particuliers dans les méthodes d'intégration numérique sur ordinateur). Il faut noter en passant que Bernoulli avait découvert ces nombres lors d'une tentative pour obtenir une formule générale qui donne la somme des puissances d'ordre p des n premiers entiers (celles des nombres eux-mêmes, de leurs carrés et de leurs cubes étaient connues depuis l'Antiquité). Les nombres de Bernoulli fournissent la somme de plusieurs séries. Comme le rapporte elle-même Ada de Lovelace — en oubliant qu'il ne s'agit que d'une égalité en valeur absolue — on peut écrire après Euler :

$$|B(2n-1)| = ((2.(2n)!)/(2\pi)^{2n}) \sum_{p=1}^{+\infty} 1/p^{2n}$$

$$\text{par exemple : } \pi^2/6 = \sum_{p=1}^{+\infty} 1/p^2,\ \pi^4/90 = \sum_{p=1}^{+\infty} 1/p^4$$

Même dans l'étude du célèbre théorème de Fermat, les nombres de Bernoulli jouent un rôle essentiel en établissant, parmi les nombres premiers, une discrimination a priori bien surprenante. On ne sera pas surpris, par contre, de les retrouver très normalement, compte tenu de leur naissance, dans les dérivées successives et les développements en série des fonctions tangente et cotangente, puisque f faisait intervenir une exponentielle. Bref, ils sont partout ; les voir se glisser dans ce qui est très probablement le premier exemple de programmation informatique jamais écrit est sans doute un signe de plus de leur étonnante versatilité et de la fascination que leurs mystères font planer sur des têtes curieuses !

riel que nous possédions — il fut déposé au musée de South Kensington en 1910 par le général Henry Babbage — ne nous permet pas de vérifier concrètement les principes de son créateur. Mais les textes de Babbage et de son fils, et surtout de Menabrea et Ada de Lovelace, sont assez significatifs.

Une formidable anticipation

Les instruments principaux étaient des cartes perforées, assemblées en longues files ou chaînes — l'équivalent de nos « listes » modernes — comme sur les métiers à tisser. Certaines portaient simplement des nombres ; ces *Variables Cards* étaient chargées de données d'entrée, de résultats ou de valeurs intermédiaires, exactement comme nos mémoires vives. Certaines définissaient les « types » des variables, comme en Fortran ou en Pascal. D'autres étaient de nature logique : parmi les *Operations Cards,* on trouvait des instructions mathématiques usuelles (+, −, ×, /) mais aussi des *Directives Cards* qui établissaient les liaisons entre les opérations, permettant par exemple de sauter sélectivement certaines cartes, d'en répéter des blocs entiers, etc.

Programmer ne pouvait évidemment se faire qu'en langage machine (ou assembleur, ce mot étant ici particulièrement adéquat). Chaque utilisateur pouvait avoir sa bibliothèque de cartes réunies une fois pour toutes pour des applications familières. Un programme de calcul de l'expression (AB + C)D, emprunté à une note de Henry Babbage, se présente ainsi :

- mettre A en mémoire 1 (dans la colonne 1) ;
- idem pour B, C, D respectivement en 2, 3, 4 ;

Quelques coefficients, A(n):

$A(0) = -1/2\ ((2n-1)/(2n+1))$
$A(1) = n$
$A(3) = (n(n-1)(2n-1))/6$

Et pour $p \leq n$:

$A(2p-1) = (2n)!/((2p)!(2n-2p+1)!)$
D'où :
$A(2p+1)/A(2p-1) = ((n-p)/(p+1)) \times ((2n-2p+1)/(2p+1))$

Et surtout, la valeur très simple :

$A(2n-1) = 1$

- faire passer A puis B du magasin au moulin ;
- calculer P = AB et le mémoriser en 5 ;
- Placer P et C dans le moulin ;
- calculer Q = P + C et le mémoriser en 6 ;
- amener Q et D dans le moulin ;
- calculer R = QD et le mémoriser en 7 ;
- imprimer R.

Tout ceci, assez banal, nécessite 17 cartes. Les textes de Menabrea et Ada de Lovelace offrent des exemples bien plus complexes. L'article principal étudie la résolution d'un système de deux équations linéaires à deux inconnues — sans s'occuper des cas exceptionnels. La traductrice s'attaquera à des niveaux de difficulté plus grands notamment sur deux points : une note consacrée à des calculs trigonométriques, et surtout à la fin des annexes, au calcul systématique des célèbres nombres de Bernoulli. Tous les exemples qu'elle développe semblent dus à son talent, sauf peut-être le dernier : Babbage affirmera plus tard lui avoir fourni une ébauche de solution, qui était malheureusement gravement inexacte. On voit que les « bogues » n'ont pas attendu notre siècle !

Quelques cartes et des calculs infinis

Les textes de l'inventeur fournissent évidemment de précieux renseignements : on y lit par exemple le principe des calculs en multiple précision, par lesquels la puissance théorique de la machine est presque illimitée. Mais le plus important à nos yeux modernes est à lire dans l'album de la comtesse. Si c'est à Babbage lui-même que l'on doit par exemple la notion de test avec bifurcations, c'est elle qui l'explique en toute clarté. Elle remarque, ou en tous cas démontre sans ambiguïté qu'un nombre fini de cartes, utilisées séquentiellement autant de fois qu'il est nécessaire, peut servir à résoudre des problèmes nécessitant des actions aussi nombreuses que l'on voudra. C'est évidemment un principe fondamental.

L'étude sur les nombres de Bernoulli est indiscutable de ce point de vue : même s'il faut théoriquement une infinité de mémoires pour emmagasiner tous les résultats intermédiaires, Ada montre avec évidence, en assumant tous les détails, que l'essentiel du calcul, très technique, ne nécessite que la manipulation de ce qu'elle appelle un « cycle » de 25 cartes, toujours les mêmes, et que 75 cartes en tout suffisent (nous dirions un programme de 75 lignes, avec une sous-routine principale de 25). Il est difficile de savoir si Ada fut réellement la première programmeuse ou si elle ne fut que l'excellente rédactrice de son maître et ami : mais elle aurait été, en tous les cas, un professeur hors classe ! A part la mention par laquelle Babbage déclare lui avoir suggéré l'étude sur Bernoulli, on ne trouvera pas trace de demande de priorité sur les idées de l'article de 1843 : il semble donc juste d'en créditer la comtesse de Lovelace pour la plus grande part.

Quelques indications biographiques sur les auteurs des articles de 1842 et 1843

Le Général du Génie Luigi Federico Menabrea, né à Chambéry le 4 septembre 1809, mourut marquis de Valdora et ancien Premier ministre italien (de 1867 à 1880), le 24 mai 1896. Il avait étudié les mathématiques à Turin, et avait été prié par un curieux personnage, le baron Giovanni Plana, polytechnicien de la promotion de 1800 et ami de Stendhal, de rédiger des notes sur les conférences qu'à sa demande Babbage avait tenues à Turin en septembre ou octobre 1840. Ses 21 pages devinrent 53 après qu'Ada (1815-1852) se fût chargée de traduire et d'étendre son travail. Elle connaissait Babbage depuis le 5 juin 1833, date que l'on peut donc considérer, au moins symboliquement, comme marquant le début de l'informatique moderne.

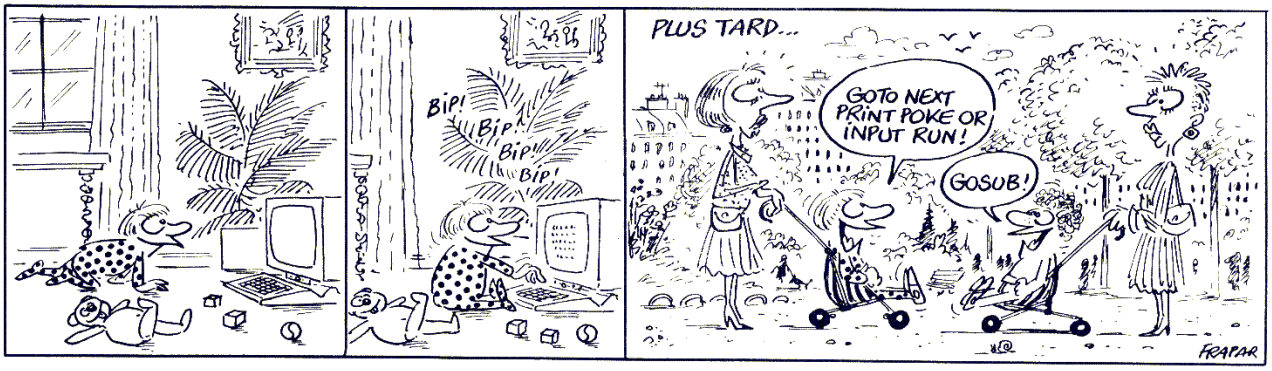
Fille d'une mathématicienne amateur d'ailleurs très douée, et du célèbre poète Lord Byron, elle fut rejetée par ce dernier qui ne voulut jamais la voir et quitta même sa mère avant sa naissance. Elle épousera en 1835 Lord King qui devint comte de Lovelace et baron Wentworth. Après son éclatante contribution au travail de Babbage — qui n'alla d'ailleurs pas sans conflits dus à des caractères affirmés de part et d'autre — sa courte vie fut très pénible. Cherchant dans le jeu, les courses de chevaux et la drogue, des sensations hors du commun, elle mourra en novembre 1852 d'un cancer de l'utérus. La terrible Lady Byron l'isola férocement durant son agonie, la privant d'opium pour la punir d'une liaison scandaleuse ayant mis en péril les bijoux de famille pour une dette d'hippodrome... Les deux êtres d'exception que côtoya cette mégère n'eurent guère à s'en féliciter !

Si vous lisez l'anglais

A un moment où l'intérêt pour l'histoire de l'informatique se développe à coups de publications introuvables et hors de prix, on se doit sans aucun doute de signaler que pour une dizaine de dollars on peut commander dans une librairie spécialiste du livre américain ou étranger, le remarquable livre de poche (Éditions Dover) sur Charles Babbage, contenant ses principaux textes et surtout l'article de Menabrea et Ada de Lovelace (sous la direction de P. et E. Morrison). Sa lecture, crayon en main, fait progresser davantage en programmation qu'une bibliothèque complète. Voilà un beau cadeau de Noël !

André WARUSFEL

"DEMAIN, L'INFORMATIQUE VA DEVENIR UNE SECONDE LANGUE POUR LES FRANÇAIS." (LAURENT FABIUS, PREMIER MINISTRE)

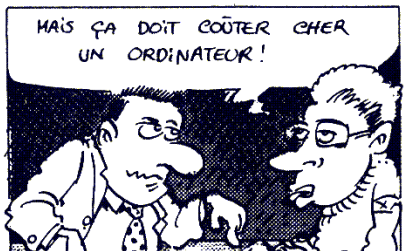


LISTER L'ERREUR AUTOMATIQUEMENT

DANS certains Basic, lors du déroulement d'un programme, s'il se produit une erreur de syntaxe, la ligne incriminée est listée automatiquement. Cette « fonction » bien intéressante pour mettre au point un programme n'existe pas sur le Canon X-07. Heureusement, il est possible d'y remédier avec un programme en langage-machine et deux lignes supplémentaires dans le programme Basic à mettre au point.

Une erreur de syntaxe est si vite arrivée ! Le Canon X-07 est capable lui aussi de lister la ligne où l'erreur s'est produite. Il lui suffit d'un programme en langage-machine et de deux lignes supplémentaires au programme Basic concerné. L'erreur peut être de type quelconque : aussi bien de syntaxe que due à une utilisation erronée des fonctions. En outre, pour en permettre une étude ou des modifications, la ligne est bloquée comme avec LIST @ .

Dans un premier temps, il faut mettre en place la routine en langage-machine à l'aide d'un programme Basic (Programme d'implantation). Le seul problème réside dans la détermination



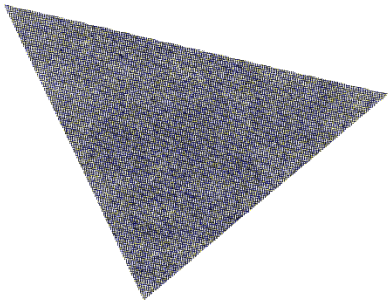
Le coin des curieux

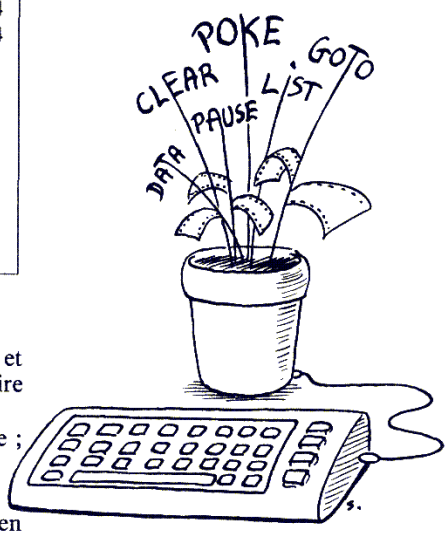
Pour ceux qui souhaitent en savoir plus, voici la liste désassemblée avec des commentaires et divers renseignements.

LD B,8	Dépilage dû à EXEC	6,8
POP HL		225
DJNZ 253		16,253
LD HL,(789)	Récupération numéro ligne erreur	42,21,3
PUSH HL		229
POP DE		209
CALL 62219	Recherche adresse ligne	205,11,243
LD IY, 16384	Pour simuler LIST @	253,33,0,64
JP 65162	Saut routine LIST	195,138,254

Quelques informations supplémentaires :

- (789) + (790)*256 : numéro de ligne où s'est produite l'erreur
- (177) : numéro de l'erreur
- (795) : indicateur erreur si 255, sinon 0
- (793) + (794)*256 : adresse réelle pour le GOTO du ON ERROR
- en 62219 : routine de recherche d'adresse de ligne contenue dans (DE), résultat dans (BC)
- en 65162 : routine LIST numéro de ligne de fin en pile, adresse première ligne dans (BC)





de l'adresse d'implantation de la routine : AD. Celle-ci est fortement liée à la configuration de chaque X-07 (pour plus de détails sur l'implantation de routines en langage-machine dans le Canon X-07, voir *LIST* 11, pages 35 à 37). Mais il est possible, quasiment sans risque, de prendre comme valeur pour AD, la taille mémoire maximum en place (8, 12, 16, 20, 24 Ko). Il faut savoir que :

- la taille de la zone fichiers est fixée par FSET ;
- la taille de la zone chaînes est fixée par CLEAR, elle est de 50 par défaut ;
- la taille de la pile système est d'environ 500 ;
- la taille du ou des programmes en langage-machine est forcément variable.

Attention aux débordements

Le choix d'une telle valeur pour AD a l'avantage de conserver la ou les routines, même après l'extinction, et de ne pas risquer d'interaction avec le Basic. Il vaut mieux tout de même vérifier la présence des routines à chaque utilisation, car un gros programme Basic ou un grand nombre de variables peuvent altérer les codes.

Un exemple de détermination d'adresse utilisable avec une configuration de 12 Ko (soit le modèle de base et une carte mémoire de 4 Ko), c'est-à-dire 12288 octets, peut être :

- 2000 octets de fichiers mémoire vive ;
- 50 octets pour les chaînes ;
- 500 octets pour la pile ;
- 200 octets pour un programme en langage-machine déjà en place et pour le suivant.

Programme d'implantation

```
10 CLS: AD = XXXXX
20 READ A : IF A < 0 THEN END
30 POKE AD,A : AD=AD+1 : GOTO 20
40 DATA 6,8,225,16,253,42,21,3,229,209
50 DATA 205,11,243,253,33,0,64,195,138,254, – 1
```

L'adresse AB doit être calculée en fonction de la configuration du Canon X-07 (voir exemple dans le texte et *LIST* 11, pages 35 à 37).

Ainsi, la valeur choisie pour l'adresse AD est : AD = 12288 – 2750 = 9538

Il est donc possible de prendre 9538 comme adresse d'implantation. Les codes langage-machine seront mis en place en 9538, puis 9539, et ainsi de suite. Cette première adresse est à retenir car les appels au programme en langage-machine y feront référence.

Quant au (ou aux) programme Basic auquel s'applique cette routine, il suffit de lui ajouter les deux lignes suivantes :

1 ON ERROR GOTO 65000:REM BRANCHEMENT EN CAS D'ERREUR

65000 EXEC AD:REM EXECUTION A L'ADRESSE D'IMPLANTATION

(avec l'exemple ci-dessus, il faut remplacer AD par 9538).

Les numéros de ligne, 1 et 65000, sont arbitraires, mais il est nécessaire de passer par le ON ERROR. Ainsi, une fois que tout est en place, si une erreur se produit pendant le déroulement du programme Basic, la ligne concernée est listée.

Laurent GRAS

VOTRE ORDINATEUR A-T-IL LA BOSSE DES MATHS ?

TOUS les Basic n'ont pas la même "puissance" mathématique. Pour mesurer cette dernière, il faut tenir compte de la précision des calculs, des intervalles de calcul et du nombre de fonctions mathématiques. La comparaison de différents Basic révèle que, dans le domaine mathématique, les ordinateurs de poche sont souvent les meilleurs.

Une première observation du tableau récapitulatif des fonctions mathématiques indique que la précision des calculs est plus importante sur les petits ordinateurs, les ordinateurs de poche : FX-702P, PC-1212, Canon X-07. Ce dernier effectue les calculs sur 14 chiffres, quel que soit le type des variables.

Le FX-702P n'est pas seulement très précis. Il se distingue aussi par le grand nombre de ses fonctions mathématiques : 60. Ce nombre tient à la présence de 17 fonctions statistiques (moyenne, écart-type, corrélation, etc.) et de 12 fonctions trigonométriques. Seuls le FX-702P et le FX-750P disposent de telles fonctions.

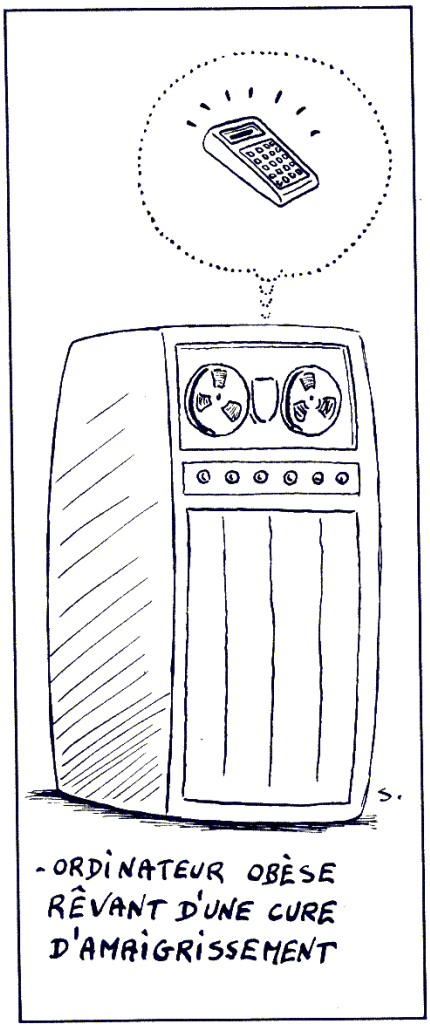
Dans tous les Basic, on trouve : les quatre opérateurs (+ − * /), les six opérateurs de relation (>, <, >=, <=, <>, =), l'élévation à la puissance, la racine carrée, la valeur absolue, la partie entière, sinus, cosinus, tangente, exponentielle, exponentielle de 10 et logarithme naturel. Ce qui fait un noyau de 20 fonctions communes.

La fonction *signe* n'est absente que du Basic du QL, la fonction *arctangente* — inverse de tangente — ne se trouve pas sur les Thomson MO5 et TO7/TO7-70, le générateur de nombres aléatoires manque aux Sharp PC-1212 et PC-1500 sur lesquels il est généralement écrit en quelques lignes de Basic.

Quant au Basic du Commodore 64, il semble bien pauvre en fonctions mathématiques. Mais il dispose d'un atout de taille : la définition de fonctions par DEFFN.

La "bosse des maths" n'est pas donnée à tous les Basic et ceux qui l'ont n'ont peut-être pas la "bosse des jeux"...

LIST



- ORDINATEUR OBÈSE RÊVANT D'UNE CURE D'AMAIGRISSEMENT

Ordinateurs	Amstrad	Atari 130 XE	Atmos	BBC	Canon X07	C.64	Dai	FX-702P	FX-750P	Lansay 64	MO 5	MSX	PB-100	PC-1212	PC-1500	QL	Spectrum	TO7/ TO7-70	ZX 81
Domaine de définition des variables entières réelles	− 32768 à 32767 ±1.7E38 à ±2.9E-39	− 32768 à 32767 ±9.999E99 à ±1E-99	− 32768 à 32767 ±1.7E38 à ±2.9E-39	2147483647 ±1.7E38 à ±2.9E-39	− 32768 à 32767 ±9.999E62 à ±9.999E-63	±1.701E38 à ±2.938E-39	-2^{31} à $2^{31}-1$ ±1E18 à ±1E-18	±9.999E99 à ±1E-99	±9.999E99 à ±1E-99	±9.999E62 à ±1E-64	− 32768 à 32767 ±1.7E38 à ±2.9E-39	−32768 à 32767 ±9.999E62 à ±1E-64	±9.999E99 à ±1E-99	±9.999E99 à ±1E-99	− 32768 à 32767 ±9.999E99 à ±1E-99	− 32768 à 32767 $\pm 10^{615}$ à $\pm 10^{-615}$	-2^{32} à $2^{32}-1$ ±1E38 à ±2.9E-39	− 32768 à 32767 ±1.7E38 à ±2.9E-39	±1.7E38 à ±2.9E-39
Précision des calculs (en double précision)	9 chiffres	9 chiffres	9 chiffres	9 chiffres	14 chiffres	9 chiffres	6 chiffres	12 chiffres	10 chiffres	9 chiffres	6 chiffres	7 chiffres (16)	12 chiffres	12 chiffres	10 chiffres	8 chiffres	8 chiffres	6 chiffres (16)	9 chiffres
Conversion de variables en entier signé en entier en simple précision en double précision	UNT CINT CREAL				CINT CSNG CDBL						CINT	CINT CSNG CDBL						CINT CSNG CDBL	
Définition de variables numériques entières simple précision double précision	DEFINT DEFREAL				DEFINT DEFSNG DEFDBL		IMP INT IMP FPT			NUMERIC	DEFINT DEFSNG	DEFINT DEFSNG DEFDBL						DEFINT DEFSNG DEFDBL	
Définition de fonction	DEFFN		DEFFN	DEFFN	DEFFN	DEFFN (X)						DEFFN					DEFFN		
Arithmétique simple 4 opérations puissance division entière reste division reste entier et positif racine carrée valeur absolue signe partie entière partie décimale nombre sans sa partie décimale entier supérieur arrondi troncature maximum minimum	+ − * / ↑ \ MOD SQR ABS SGN INT ROUND FIX MAX MIN	+ − * / ∧ SQR ABS SGN INT	+ − * / ↑ SQR ABS SGN INT	+ − * / ∧ DIV MOD SQR ABS SGN INT	+ − * / ∧ ¥ MOD SQR ABS SGN INT FIX	+ − * / ↑ SQR ABS SGN INT	+ − * / ↑ MOD SQR ABS SGN INT FRAC	+ − * / ↑ SQR ABS SGN INT FRAC RND	+ − * / ∧ SQR ABS SGN INT FRAC ROUND	+ − * / ∧ REM MOD SQR ABS SGN INT FP IP CEIL ROUND TRUNCATE MAX MIN	+ − * / ∧ @ MOD SQR ABS SGN INT FIX	+ − * / ∧ \ MOD SQR ABS SGN INT FIX	+ − * / ↑ SQR ABS SGN INT FRAC RND	+ − * / ∧ √‾ ABS SGN INT	+ − * / ∧ √‾ ABS SGN INT	+ − * / ∧ DIV MOD SQRT ABS INT	+ − * / ↑ SQR ABS SGN INT	+ − * / ∧ @ MOD SQR ABS SGN INT FIX	+ − * / ** SQR ABS SGN INT FRAC
Opérateurs et fonctions logiques 6 opérateurs de relation et ou ou exclusif négation équivalence implication et binaire ou binaire ou exclusif binaire non binaire décalage binaire	oui AND OR XOR NOT	oui AND OR NOT	oui AND OR NOT AND ORA EOR	oui AND OR NOT	oui EQV AND OR XOR NOT	oui AND OR NOT	oui AND OR IAND IOR IXOR INOT SHL, SHR	oui	oui	oui AND OR BAND BOR	oui AND OR XOR NOT EQV IMP	oui AND OR XOR NOT EQV IMP	oui	oui	oui AND OR NOT	oui AND OR XOR NOT && || ^^ ~~	oui AND OR NOT	oui AND OR XOR NOT EQV IMP	oui AND OR NOT

Ordinateurs	Amstrad	Atari 130 XE	Atmos	BBC	Canon X07	C.64	Dai	FX-702P	FX-750P	Lansay 64	MO 5	MSX	PB-100	PC-1212	PC-1500	QL	Spectrum	TO7/ TO7-70	ZX 81
Fonctions trigonométriques et angulaires																			
sinus	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN
cosinus	COS	COS	COS	COS	COS	COS	COS	COS	COS	COS	COS	COS	COS	COS	COS	COS	COS	COS	COS
tangente	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN
cotangente										COT						COT			
arcsinus				ASN			ASIN	ASN	ASN	ASIN			ASN	ASN	ASN	ASIN	ASN		ASN
arcosinus				ACS			ACOS	ACS	ACS	ACOS			ACS	ACS	ACS	ACOS	ACS		ACS
arctangente	ATN	ATN	ATN	ATN	ATN	ATN	ATN	ATN	ATN	ATN		ATN	ATN	ATN	ATN	ATAN	ATN		ATN
arccotangente																ACOT			
sinus hyperbolique								HSN	HYPSIN	SINH									
cosinus hyperbolique								HCS	HYPCOS	COSH									
tangente hyperbolique								HTN	HYPTAN	TANH									
inverse sinus hyperbolique								AHS	HYPASN										
inverse cosinus hyperbolique								AHC	HYPACS										
inverse tangente hyperbolique								AHT	HYPATN										
sécante										SEC									
cosécante										CSC									
pi	PI		PI	PI		"	PI	"	PI	PI			"	"	" ou PI	PI	PI		PI
option mesures d'angles								Mode 4,5,6	Angle 0,1,2	OPTION			Mode 4,5,6						
conversion en degrés	DEG	DEG		DEG						DEG				DEG	DEG	DEG			
conversion en radians	RAD	RAD		RAD						RAD				RAD	RAD	RAD			
conversion en grades														GRAD	GRAD				
calculs en	deg ou rad	deg ou rad	rad	deg ou rad	rad	rad	rad	rad,deg ou grad	rad,deg ou grad	deg ou rad	rad	rad	deg, rad ou grad	deg, rad ou grad	deg, rad ou grad	deg ou rad	rad	rad	rad
Fonctions logarithmes																			
exponentielle	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP
E, puissance de 10	E	E	E	E	E ou D	E	E	E	E	E	E	E ou D	E	E	E	E	E	E ou D	E
log népérien	LOG	LOG	LN	LN	LOG	LOG	LOG	LN	LOG	LOG	LOG	LOG	LN	LN	LN	LN	LN	LOG	LN
log décimal	LOG 10	C LOG	LOG	LOG			LOGT	LOG	LGT	LOG10			LOG	LOG	LOG	LOG10			
log base 2										LOG 2									
inverse du log népérien							ALOG												
Les bases																			
calculs hexa	&H ou &				&H		#					&H				&			
calculs binaires	&X											&B					BIN		
calculs en octal					&O ou &							&O							
Conversion d'une base à l'autre																			
binaire - décimal	STR$(&X									BIN									
hexadécimal - décimal	STR$(&H						PRINT #									& (x)			
décimal - binaire	BIN$											BIN$							
décimal - hexadécimal	HEX$		HEX$		HEX$		HEX$		HEX$			HEX$						HEX$	
décimal - octal												OCT$						OCT$	
sexagésimal - décimal								DEG	DEG						DEG				
décimal - sexagésimal								DMS	DMS$					DMS	DMS				
Autres fonctions scientifiques																			
factorielle								!											
conversion polaire-rectangulaire								PRC											
conversion rectangulaire-polaire								RPC											
fonctions statistiques								17	15										
générateur aléatoire	Randomize						interne			Randomize						Randomise	Randomize		RAND
générateur d'entiers	RND									RND (X)						RND (X)			
générateur de réels entre 0 et 1		RND (X)	RND	RND	RND	RND	RND (1)	RAN #	RND	RND	RND	RND	RAN #			RND	RND	RND	RND
l'infini										INF									
epsilon										EPS									
Nombre de fonctions	50	29	33	35	42	28	44	60	56	56	34	46	32	30	35	42	32	40	30

LES POINTEURS FONT DE BELLES LISTES

LES pointeurs, l'ordinateur ne pourrait pas s'en passer. Dans Prolist, un programme qui liste les programmes, on va pouvoir en observer quelques-uns à l'œuvre. Des exemples d'exécution vous montreront la supériorité du produit : vous choisissez la largeur de la ligne imprimée, le nombre de lignes imprimées par page, la marge et le titre. Les caractères de contrôle de chez Commodore sont listés en clair – parfaitement ! – en anglais pour faire comme sur les touches, mais si vous voulez modifier ça, ne vous en privez surtout pas.

Le programme Prolist fait appel — discrètement, il est vrai — aux pointeurs du Basic. Contentons-nous d'observer leur travail sans les approcher de trop près, de crainte de les effrayer. Prolist est un programme qui permet de lister des programmes avec une mise en page choisie par l'utilisateur. Voyons comment on s'en sert.

Ce programme est numéroté « gros » : de 63450 à 63500. C'est qu'il est destiné à se placer à la suite du programme dont vous voulez une liste haut de gamme. Il doit être lancé par un RUN 63450. Si vous disposez d'un utilitaire Merge ou Append, pas de problème. Sinon, réjouissez-vous, nous allons apprendre à faire un Append à la main.

La recette de l'Append à la main tient en quelques lignes :

- mettre en mémoire le premier programme (celui dont tous les numéros de lignes sont inférieurs au plus petit numéro de ligne du second) ;
- frapper, en mode direct, A = PEEK (45) + 256*PEEK(46) – 2:PRINT A – INT(A/256)*256,INT(A/256). On obtient deux valeurs ;
- frapper POKE 43,*première valeur*: POKE 44,*deuxième valeur*:CLR, puis la touche RETURN ;
- charger le second programme, Prolist par exemple ;
- quand le curseur réapparaît, frapper, toujours en mode direct, POKE 43,1: POKE 44,8.

Ouvrir le canal de l'imprimante

Maintenant, on peut désosser le programme. Première ligne, OPEN 4,4. Après s'être baladée pas mal au cours de l'élaboration du programme, la commande d'ouverture du canal de l'imprimante s'est retrouvée tout au début. La ligne 63456 dimensionne deux tableaux. Le premier, T$(n), recevra la première série de DATA : les mots clés de Basic. Le second, Q$(n), recevra les caractères de contrôle du C.64.

Mais il y a un problème : ce sous-programme de listage formaté est fait pour être ajouté par Append au programme à lister. Sinon, il n'aurait d'autre utilité que de se lister lui-même, ce qui serait décoratif, certes, mais assez limité. Or, il y a ces deux tableaux de DATA à lire. Et si le programme à lister contient lui-même des DATA, ils risquent de prendre, dans le meilleur des cas, la place des mots clés de Basic, et

LES POINTEURS FONT DE BELLES LISTES

Prolist
Programme pour
Commodore 64
Auteur François J. Bayard
Copyright *LIST* et l'auteur

de donner de drôles de résultats... D'où une ruse, en l'absence de RESTORE N : lire des DATA dans une boucle infinie ou presque, jusqu'à ce qu'on rencontre celui qu'on veut. C'est le but de la fausse boucle de la ligne 63456 : elle cherche les trois étoiles qu'on retrouve comme premier DATA à la ligne 63489.

Le vocabulaire Basic est donc lu. Puis quelques cases du tableau Q$(n), celles qui correspondent à des caractères de contrôle à lister en clair, sont remplies avec une technique différente : au lieu de se servir d'un compteur de boucle comme indice, on lit d'abord l'indice, puis, s'il est différent de zéro, le contenu. C'est pour cela que le dernier DATA de la ligne 63500 est un zéro.

A plusieurs reprises au cours du programme, nous aurons à lire en mémoire un nombre stocké sur deux octets. On sait que la recette en pareil cas est de lire le contenu du premier octet, puis d'y ajouter le contenu du second multiplié par 256. On gagnera de la place en définissant une fonction au moyen de l'ins-

```
PROLIST-64                          PAGE 1

63450 REM ************************
63451 REM *     P R O L I S T    *
63452 REM * (C)1985 F.J. BAYARD *
63453 REM *      ET  L I S T     *
63454 REM ************************
63455 OPEN4,4
63456 DIMT$(128),Q$(255):FORI=1TO9E9:READ←
      X$:IFX$<>"***"THENNEXT
63457 FORI=1TO91:READT$(I):NEXT
63458 READV:IFVTHENREADX$:Q$(V)=X$:GOTO63←
      458
63459 DEFFNA(X)=PEEK(X)+256*PEEK(X+1):A=F←
      NA(43):REM CBM:FNA(40),VIC/64:FNA(4←
      3)
63460 INPUT"DEBUT,FIN ";D,F:INPUT"TITRE";←
      T$:INPUT"LIGNES PAR PAGE";LP
63461 INPUT"MARGE";M:M$="":FORI=1TOM:M$=M←
      $+CHR$(32):NEXT
63462 INPUT"LARGEUR DE LIGNE IMPRIMEE";LL←
      :GOSUB 63487
63463 X=FNA(A):L=FNA(A+2):G=0:IFX=0ORL>FT←
      HENPRINT#4:CLOSE4:END
63464 IFL<0THENA=X:GOTO63463
63465 GOSUB63483:CC=0:PRINT#4,M$;:X$=RIGH←
      T$("[3SPACES]"+STR$(L),5)+" ":GOSUB←
      63477:R=1
63466 FORK=A+4TOA+93:P=PEEK(K):W=PEEK(K+1←
      ):IFP=0THENPRINT#4:A=X:GOTO63463
63467 IFP=34THENG=NOTG
63468 IFGTHENGOSUB63471:NEXT
63469 IFNOTGANDP>127THENX$=T$(P-127):GOSU←
      B63477:NEXT
63470 PRINT#4,CHR$(P);:GOSUB63480:CC=CC+1←
      :NEXT
63471 IFW=PAND(P<33OR(P>127ANDP<161))THEN←
      R=R+1:RETURN
63472 IFQ$(P)<>""THEN63474
63473 PRINT#4,CHR$(P);:GOSUB63480:CC=CC+1←
      :RETURN
63474 IFP=32ANDR=1THEN63473
63475 X$=MID$(STR$(R),2):IFR<2THENX$=""
63476 X$="["+X$+Q$(P)+"]":GOSUB63478:R=1:←
      RETURN
63477 IFX$=""THENRETURN
63478 FORC=1TOLEN(X$):PRINT#4,MID$(X$,C,1←
      );:GOSUB63480
63479 CC=CC+1:NEXTC:RETURN
63480 IFCC>=LLANDW=0ANDC=LEN(X$)+1THENRET←
      URN
63481 IFCC>=LLTHENPRINT#4,CHR$(95):GOSUB6←
      3483:PRINT#4,M$;SPC(6);:CC=5
63482 RETURN
63483 CL=CL+1:IFCL<=LP THEN RETURN
63484 PRINT"[DOWN][RVS]S[OFF]UITE ?"
63485 GETR$:IFR$<>"S"THEN63485
63486 PRINT"[CLR]IMPRESSION..."
63487 NP=NP+1:Z=CC:CC=0:PRINT#4,M$;:X$=T$←
      :GOSUB63477:PRINT#4,SPC(17);"PAGE "←
```

truction DEF FNA(X). On va d'ailleurs s'en servir tout de suite pour aller chercher l'adresse du début de Basic que l'ordinateur garde... grâce à un pointeur. Et si vous vous demandez pourquoi ne pas donner cette adresse directement et en clair, c'est que vous ne travaillez pas sur un Vic 20, dont le Basic se balade en fonction de l'extension de mémoire branchée ! Ce pointeur est en 43-44 sur Vic et C.64, en 40-41 sur Pet/CBM.

Le programme demande ensuite les numéros de la première et de la dernière lignes à lister. Ces numéros ne sont pas critiques en ce sens que vous pouvez répondre 0,63499 (il s'agit d'un INPUT double : deux entrées séparées par une virgule) même si votre programme ne comporte ni ligne 0, ni ligne 63499.

Ceux qui n'ont jamais retrouvé dans leurs papiers une feuille de programme dont rien ne permettait de savoir à quoi il servait ne comprendront pas pourquoi Prolist prévoit de sortir un titre en haut de chaque nouvelle page. Les autres si. Le nombre de lignes par page imprimée est également paramétrable, ainsi que la marge (qui sera simplement imprimée au début de chaque ligne sous forme d'une chaîne de blancs M$), et la largeur de la ligne imprimée.

```
PROLIST-64                    PAGE 2

        ;
63488 CL=2:X$=MID$(STR$(NP),2):GOSUB63477←
      :CC=2:PRINT#4:PRINT#4:RETURN
63489 DATA***,END,FOR,NEXT,DATA,INPUT#,IN←
      PUT,DIM,READ,LET,GOTO,RUN,IF,RESTOR←
      E
63490 DATAGOSUB,RETURN,REM,STOP,ON,WAIT,L←
      OAD,SAVE,VERIFY,DEF,POKE,PRINT#,PRI←
      NT
63491 DATACONT,LIST,CLR,CMD,SYS,OPEN,CLOS←
      E,GET,NEW,TAB(,TO,FN,SPC(,THEN,NOT
63492 DATASTEP,+,-,*,/,↑,AND,OR,>,=,<,SGN←
      ,INT,ABS,USR,FRE,POS,SQR,RND,LOG,EX←
      P
63493 DATACOS,SIN,TAN,ATN,PEEK,LEN,STR$,V←
      AL,ASC,CHR$,LEFT$,RIGHT$,MID$,GO
63494 DATACONCAT,DOPEN,DCLOSE,RECORD,HEAD←
      ER,COLLECT,BACKUP,COPY,APPEND,DSAVE
63495 DATADLOAD,CATALOG,RENAME,SCRATCH,DI←
      RECTORY
63496 DATA 5,WHT,8,COFF,9,CON,14,TXT,17,D←
      OWN,18,RVS,19,HOME,20,DEL,28,RED
63497 DATA29,RIGHT,30,GRN,31,BLU,129,ORG,←
      133,F1,134,F3,135,F5,136,F7,137,F2
63498 DATA138,F4,139,F6,140,F8,142,GRAPH,←
      144,BLK,145,UP,146,OFF,147,CLR
63499 DATA148,INST,149,BRWN,150,L.RED,151←
      ,GR.1,152,GR.2,153,L.GRN,154,L.BLU
63500 DATA155,GR.3,156,PUR,157,LEFT,158,Y←
      EL,159,CYN,160,SH.SP,0
```

Les caractères de contrôle du Pet/CBM

ASCII		CARA.	MAJ/GR	MAJ/MIN	
3	#	STOP			
7	*	BELL			SONNERIE
9	*	TAB			APPEL TABULATION
13		CR			RETOUR-CHARIOT
14	*	TEXT			MODE TEXTE
16	*	CFC			EFFACEMENT DEPUIS LE CURSEUR
17		DOWN			CURSEUR BAS
18		RVS			INVERSION VIDEO
19		HOME			CURSEUR EN HAUT A GAUCHE (ECRAN)
19		HOME			DESACTIVE LA PAGINATION (IMPRIMANTE)
20		DEL			EFFACEMENT DE CARACTERE
29		RIGHT			CURSEUR DROIT
131	#	RUN			
135	*	2BELL			DOUBLE SONNERIE
137	*	SETTAB			POSE D'UN TAQUET DE TABULATION
141		SH.CR			SHIFT RETOUR-CHARIOT
142	*	GRAPH			PASSAGE EN MODE GRAPHIQUE
145		UP			CURSEUR HAUT
146		OFF			FIN D'INVERSION VIDEO
147		CLR			EFFACEMENT D'ECRAN
147		CLR			PAGINATION ACTIVE (IMPRIMANTE)
148		INST			INSERTION
150	*	CTC			EFFACEMENT JUSQU'AU CURSEUR
157		LEFT			CURSEUR GAUCHE
160		SH.SP			SHIFT/ESPACE

INUTILISE EN PRINT
* CBM 4000-GE/8000 SEULEMENT

Les lignes de Commodore

Les initialisations terminées, un appel au sous-programme en 63487 imprime le titre du programme et le numéro de page, saute une ligne, met à 2 le compteur de lignes CL, sur quoi on peut passer aux choses sérieuses. Il faut savoir qu'une ligne de Basic chez Commodore commence par deux octets de *lien*, qui contiennent l'adresse du début de la ligne suivante. On trouve ensuite le numéro de ligne, toujours codé sur deux octets, puis la ligne elle-même, et enfin un zéro qui indique la fin de la ligne. Et puis ça recommence : adresse de début de la ligne suivante, numéro, corps de la ligne, zéro, etc. Si la ligne est la dernière du programme, ce zéro est suivi de deux autres zéros, ou si vous préférez, l'adresse de la ligne suivante est zéro.

Lorsque la ligne 63463 est exécutée pour la première fois, la variable A contient l'adresse du début de la première ligne du programme à lister (2049 sur le C.64). A et A + 1 contiennent donc l'adresse du début de la ligne suivante, extraite par FNA(A) et rangée dans la variable X. A + 2 et A + 3 contiennent le numéro de ligne, extrait par FNA(A + 2) et rangé dans la variable L. La variable G est un drapeau qui nous permettra de savoir si les guillemets sont ouverts ou non.

Si l'adresse de la ligne suivante est zéro, ou si le numéro de la ligne s'annonce supérieur à la dernière ligne à lister, c'est que le programme a fini : il ferme poliment le canal imprimante derrière lui et va se coucher. En revanche, si le numéro de la ligne s'annonce inférieur à la première ligne à lister, on va examiner directement la ligne suivante,

LES POINTEURS FONT DE BELLES LISTES

dont l'adresse de début était en X, mais va venir en A (ligne 63464).

La ligne 63465 commence par un petit crochet en 63483 pour mettre à jour le compteur de lignes imprimées, met à zéro le compteur de caractères, sort la marge, écrit le numéro de ligne sur cinq colonnes justifiées à droite, et met à un le compteur de répétitions.

Cinq caractères, un seul code

Un mot sur le sous-programme qui commence à la ligne 63477. On lui donne une chaîne de caractères X$ et il l'écrit, caractère par caractère, avec un détour via 63480 qui compte ces caractères, et compare le résultat avec la largeur de ligne LL fixée par vous. Si ce nombre dépasse LL, il imprime un CHR$(95), c'est-à-dire une flèche à gauche pour marquer que la ligne est coupée, et passe à la ligne. Il vérifie qu'il n'y a pas de changement de page à effectuer (c'est le GOSUB 63483), refait une marge, saute six espaces pour bien mettre en valeur les numéros de lignes, remet à jour le compteur de caractères et revient. Vous pouvez (ligne 63481) supprimer le SPC(6) et mettre CC = 0. Le résultat est moins clair, mais on gagne de la place. A vous de choisir.

C'est à la ligne 63466 que commence le décorticage de la ligne Basic proprement dite. En principe, une ligne de Basic chez Commodore ne dépasse pas 80 caractères, mais basta, ne soyons pas mesquin, mettons-en un chouya de plus et allons jusqu'à 93, pour rire ! C'est donc encore une fausse boucle (on n'ira jamais jusqu'au bout) qui examine un par un les octets d'adresse K et les appelle P. Pendant qu'on y est, un coup d'œil prévoyant sur l'octet suivant, baptisé W. Si l'octet en cours vaut zéro, c'est que la ligne est terminée. Dans ce cas, on passe à la ligne, physiquement sur le papier, et on passe à la ligne (de Basic) suivante.

Vient alors, à la ligne 63467, la question cruciale des guillemets. Lorsque la machine interprète une ligne de Basic, de deux choses l'une : ou bien des guillemets ont été ouverts — et non refermés — et dans ce cas la machine transmet la chaîne entre guillemets sans chercher à comprendre ni même à savoir ce qu'il y a dedans, ou bien les guillemets ne sont pas ouverts, et dans ce cas-là, il faut savoir démêler les variables, les nombres, les signes, les mots Basic. Et c'est là qu'interviennent plus précisément les mots clés du Basic.

Quand sur votre écran vous lisez PRINT, vous ne pouvez pas trouver à l'emplacement correspondant en mémoire le P, le R, le I, le N et le T. A la place de ces *cinq* caractères, l'ordinateur en a mis *un seul,* qui constitue un code, ici 153. C'est pour cela que le programme va chercher dans la liste des DATA l'inscription en clair qui correspond à ce code. Le premier mot clé Basic de nos DATA, qui est END (humour américain), a pour code 128.

Prenons un exemple, vous êtes en train de déchiffrer une ligne, et vous tombez inopinément sur le code 165. Qu'est-ce donc ? Pour le savoir, vous consultez discrètement votre drapeau guillemets :

- si les guillemets sont ouverts, ce caractère de code supérieur à 127 ne peut être qu'une majuscule ou un caractère graphique ;
- si les guillemets ne sont pas ouverts, ce caractère de code supérieur à 127 ne peut être qu'un mot clé de Basic (ligne 63469).

De même, entre guillemets, un caractère de code inférieur à 128 est une let-

Les caractères de contrôle du C.64

ASCII	CLAVIER	CARA.	MAJ/GR	MAJ/MIN	
3		STOP			(INUTILISE EN PRINT)
5	CTRL/2	WHT			BLANC
8		COFF			DESACTIVE LA TOUCHE COMMODORE
9		CON			REACTIVE LA TOUCHE COMMODORE
13	RETURN	CR			RETOUR-CHARIOT
14		TEXT			MODE TEXTE
17	CRSR↑	DOWN			CURSEUR BAS
18	CTRL/9	RVS			INVERSION VIDEO
19	HOME	HOME			CURSEUR EN HAUT A GAUCHE (ECRAN)
19	HOME	HOME			PAGINATION DESACTIVEE (IMPRIMANTE)
20	DEL	DEL			EFFACEMENT DE CARACTERE
28	CTRL/3	RED			ROUGE
29	CRSR←	RIGHT			CURSEUR DROIT
30	CTRL/6	GRN			VERT
31	CTRL/7	BLU			BLEU
129	C/1	ORG			ORANGE
131		RUN			(INUTILISE EN PRINT)
141	SH/RETURN	SH.CR			SHIFT RETOUR-CHARIOT
142		GRAPH			PASSAGE EN MODE GRAPHIQUE
144	CTRL/1	BLK			NOIR
145	SHIFT↑	UP			CURSEUR HAUT
146	CTRL/0	OFF			FIN D'INVERSION VIDEO
147	SHIFT/HOME	CLR			EFFACEMENT D'ECRAN
147	SHIFT/HOME	CLR			PAGINATION ACTIVE (IMPRIMANTE)
148	SHIFT/DEL	INST			INSERTION
149	C/2	BRWN			MARRON
150	C/3	L.RED			ROUGE CLAIR
151	C/4	GR.1			GRIS 1
152	C/5	GR.2			GRIS 2
153	C/6	L.GRN			VERT CLAIR
154	C/7	L.BLU			BLEU CLAIR
155	C/8	GR.3			GRIS 3
156	CTRL/5	PUR			VIOLET
157	SHIFT/CRSR	LEFT			CURSEUR GAUCHE
158	CTRL/8	YEL			JAUNE
159	CTRL/4	CYN			CYAN
160	SHIFT/SP	SH.SP			ESPACE SHIFTE

C : TOUCHE COMMODORE EN BAS A GAUCHE DU CLAVIER

Les caractères de contrôle qui ne figurent pas au clavier peuvent être obtenus par la séquence : [INST] [SPACE] [LEFT] [RVS] suivie du caractère correspondant. Par exemple, pour passer en mode texte, frapper : 100 PRINT BONJOUR puis [RETURN]. Ramener le curseur sur le B et frapper [INST] [SPACE] [LEFT] [RVS], puis la touche N, puisque le caractère de contrôle du mode texte apparaît dans une chaîne comme un N non shifté en vidéo inversée.

tre normale, et donc dépourvue d'intérêt ; hors guillemets, c'est un nom de variable, et il faudra en faire quelque chose.

Si donc, à la ligne 63467, P vaut 34, c'est qu'on est en présence d'un guillemet, et l'on bascule le drapeau, c'est-à-dire qu'on le met à -1 s'il valait zéro et inversement. Si l'on est en mode guillemets, on appelle le programme en 63471.

Le premier test consiste à vérifier s'il s'agit d'un caractère de contrôle (ou d'un espace) et si le suivant lui est identique. Car le programme, s'il y a six espaces, par exemple, écrira 6 SPACES.

Lorsque deux caractères de contrôle se suivent, on se borne à incrémenter le compteur de répétition R, et l'on va voir le caractère suivant. Si ce n'est pas un caractère de contrôle répété, c'est peut-être un caractère de contrôle isolé, ou le dernier d'une suite de caractères répétés. Dans ce cas, il a sa traduction dans le tableau Q$(n) et l'on passe à 63474, qui imprimera un espace si c'en est un tout seul, sortira la traduction en clair du caractère avec, éventuellement, le nombre de répétitions, et remettra le compteur de répétitions à 1 avant de revenir.

Incrémenter avec prudence

Si c'est un caractère normal, on l'imprime normalement (ligne 63473), on vérifie que l'on n'a pas atteint la fin d'une ligne imprimée, on incrémente le compteur de caractères et on passe au suivant.

La routine en 63483 incrémente le compteur de lignes préposé au changement de page. Dans cette version du programme, elle joue la prudence : en fin de page, elle arrête l'impression et demande la permission de continuer. L'utilisateur avisé n'accordera cette permission qu'après avoir mis une nouvelle feuille en place. Si l'on utilise du papier continu, on peut remplacer les lignes 63484 et 63485 par une boucle de PRINT#4 correspondant à la longueur de la page utilisée (souvent 66 ou 72) moins la longueur de page LP donnée par l'utilisateur (63460). Le programme peut alors incrémenter le compteur de pages NP, mettre de côté le compteur de caractères, inscrire en haut de la nouvelle page la marge M$, le titre du programme T$ et le numéro de page, remettre à jour le compteur de lignes CL, récupérer la valeur du compteur de caractères, et sauter proprement une ligne avant de continuer.

Les modifications à apporter pour faire tourner ce programme sur un CBM sont minimes : elles consistent simplement à adapter en 63459 l'adresse du pointeur de début de Basic, et en 63496 et suivantes, les caractères de contrôle.

Les modifications pour faire tourner le programme avec une Brother EP-44 et un Vic 20 ou un C.64 muni de l'interface Vic 1011A se répartissent en deux catégories : d'une part les lignes 63476 et 63481 qui remplacent les crochets signalant un caractère de contrôle par des parenthèses et la flèche gauche des lignes coupées par un tilde ; d'autre part celles qui ont trait au fonctionnement particulier du port RS-232C sur le C.64 (voir encadrés).

Au nombre des exotismes les plus vicieux de chez Commodore, figure la gestion du port RS-232C avec l'interface idoine. L'ouverture d'un canal RS-232C provoque l'attribution de deux zones tampons destinées aux caractères émis et reçus. Ce ne serait pas un mal en soi si, pour bien fixer les limites de ces deux zones, le C.64 n'avait besoin de réinitialiser les pointeurs (encore eux) avec un CLR bien senti. En d'autres termes, dès qu'un canal RS-232 est ouvert, on perd toutes les variables qu'on pouvait avoir en mémoire. La prudence la plus élémentaire invite donc à ouvrir son canal avant d'avoir eu la moindre variable. Quant à la fermeture, elle ne s'occupe pas de savoir si les caractères transmis sont allés jusqu'au bout. C'est le genre guichet de sécu : « Six heures moins le quart, le temps qu'on ferme, il sera six heures, repassez lundi. » D'où le petit ajout de la fin, demandant à l'utilisateur de ne fermer que quand tout le listage sera terminé.

François J. BAYARD

Modifications du programme pour la Brother EP-44

```
63455 OPEN4,2,2,CHR$(2)+CHR$(1)

      IL FAUT QUE L'OUVERTURE DU CANAL
      SE FASSE AU DEBUT DU PROGRAMME,
      CAR ELLE PROVOQUE UN CLR, C'EST-
      A DIRE LA PERTE DES VARIABLES.

63463 X=FNA(A):L=FNA(A+2):G=0:IFX=OORL>FT~
      HEN63501

      VOIR 63501 SQ.

63476 X$=CHR$(123)+X$+Q$(P)+CHR$(125):GOS~
      UB63478:R=1:RETURN

      CHR$(123) ET CHR$(125) SONT LES
      PARENTHESES: { }.

63481 IFCC LLTHENPRINT#4,CHR$(126):GOSUB~
      63483:PRINT#4,SPC(6);:CC=5

      CHR$(126) EST LE TILDE: ~

63501 PRINT"{DOWN}FRAPPER {RVS}RETURN{OFF~
      } SI L'IMPRESSION EST","TERMINEE"
63502 GETR$:IFR$<>CHR$(13)THEN63502
63503 PRINT#4:CLOSE#4:END

      LA FERMETURE DU CANAL RS232,
      COMME SON OUVERTURE, PROVOQUE
      UN CLR. IL FAUT DONC S'ASSURER
      QUE TOUTES LES DONNEES ONT ETE
      TRANSMISES AVANT DE LE FERMER.
```

Modifications du programme pour le Pet/CBM

```
63459 DEFFNA(X)=PEEK(X)+256*PEEK(X+1):A=F←
      NA(40):REM CBM:FNA(40).VIC/64:FNA(4←
      3)
63496 DATA7,BELL,9,TAB,14,TXT,16,CFC,17,D←
      OWN,18,RVS,19,HOME,20,DEL,29,RIGHT
63497 DATA32,SPACES,135,DBELL,137,SETTAB,←
      141,SH,CR,142,GRPH
63498 DATA145,UP,146,OFF,147,CLR,148,INST←
      ,150,CTC,157,LEFT,160,SH,SP,255,PI,←
      0
```

Un exemple avant et après exécution

```
AVANT :

 100 PRINT"[CLR][6DOWN]                    [DOWN]←
                                  [RVS]LISTOMANIA■"
APRES :

     LISTOMANIA                    PAGE 1

       100 PRINT"[CLR][6DOWN][22SPACES]_____[DOWN]←
           [10LEFT][RVS]LISTOMANIA[OFF]"
```

LES INCONNUES DES SYSTÈMES LINÉAIRES

En moins de quinze lignes et en un temps record, un programme Basic retrouve toutes les inconnues – quand elles existent – d'un système d'équations linéaires. Si ce programme semble répéter quatre fois la même chose, il n'en donne pas moins les bons résultats.

L'*Homo Informathematicus*, cette espèce en danger d'apparition, serait sans doute séduit par le programme *Résolution de systèmes linéaires* qui répète quatre fois le même thème, avec de subtiles variations, pour aboutir à la résolution rapide de systèmes d'équations linéaires par la méthode de Crout. Cette méthode repose sur la factorisation d'une matrice carrée en deux matrices triangulaires, « droite » et « gauche ». Nous nous serions bien gardé de l'infliger à tout autre qu'à des passionnés de calcul matriciel s'il n'avait offert quelques « plus » dignes d'attention.

Tout d'abord il est écrit en Basic standard et ne consomme qu'un minimum de mémoire (voir son DIM). Il permet donc de tirer le maximum des micros les plus modestes.

Ensuite, la restructuration de la matrice des coefficients intervient avant la saisie des termes constants, pour lesquels on dispose souvent de nombreuses séries de valeurs. Chaque entrée d'une de ces séries déclenche un calcul instantané des valeurs correspondantes prises par les inconnues.

Tout simplement fulgurant

Enfin, cette instantanéité est obtenue dès le premier calcul. Même face à un système de plusieurs dizaines d'équations à autant d'inconnues, le programme ne s'accorde jamais aucun délai de réflexion appréciable. Tous ceux qui se sont attaqués, avec des programmes classiques, à des systèmes d'équations un peu étoffés, même sur des micros relativement rapides, reconnaîtront là une performance insolite.

Certaines ombres au tableau doivent néanmoins être signalées.

- Le programme demande l'introduction des coefficients dans un ordre inhabituel qui impose une certaine attention. Il commence par les coefficients de la première ligne, puis le reste de la pre-

Résolution de systèmes linéaires
Programme en Basic
Auteur Pierre Barnouin
Copyright *LIST* et l'auteur

```
10 INPUT "degré";N:DIM A(N,N),B(N)
15 FOR K=1 TO N:FOR J=K TO N:PRINT K"-"J;
20 C=0:FOR I=1 TO K-1:C=C+A(I,J)*A(K,I):NEXT
25 INPUT"";A:A(K,J)=A-C:NEXT:PRINT
28 IF K+1>N GOTO 45
30 FOR I=K+1 TO N:PRINT I"!"K;
35 C=0:FOR J=1 TO K-1:C=C+A(I,J)*A(J,K):NEXT
40 INPUT A:A(I,K)=(A-C)/A(K,K):NEXT I,K
45 PRINT:FOR I=1 TO N:PRINT"b"I;
50 C=0:IF I>1 THEN FOR J=1 TO I-1:C=C+A(I,J)*B(J):NEXT
55 INPUT B:B(I)=B-C:NEXT
60 FOR I=N TO 1 STEP -1:PRINT"X"I"=";
65 C=0:IF I<N THEN FOR J=I+1 TO N:C=C+A(I,J)*B(J):NEXT
70 B(I)=(B(I)-C)/A(I,I):PRINT B(I),:NEXT:GOTO 45
```

Bizarres, les *PRINT* ? Et pourtant, ils passent très bien sur notre PX-8, et même sur notre TRS-80.

mière colonne ; le reste de la deuxième ligne, puis le reste de la deuxième colonne, etc. Il s'efforce de minimiser cet inconvénient en guidant la saisie par l'affichage des indices séparés par un signe approprié, " – " pour les lignes et "!" pour les colonnes. En pratique, on s'y fait assez vite, d'autant plus facilement que les entrées se succèdent tantôt en ligne, tantôt en colonne sur l'écran.

- Les valeurs prises par les inconnues sortent dans l'ordre inverse de celui adopté pour écrire chaque équation. Il est facile d'éliminer cet inconvénient — si on considère que c'en est un — au prix d'un léger temps mort. Un tel « inconvénient » est conservé pour mieux faire ressortir la rapidité du programme. L'affichage des indices écarte, bien entendu, toute ambiguïté.
- Tous les Basic ne disposent pas de la double précision. C'est pourquoi elle n'apparaît pas dans ce programme. Ceux qui ont le choix pourront ajouter l'instruction DEFDBL A-C en tête de programme.
- Enfin, si par une grande malchance, le « pivot » de la matrice des coefficients devait être nul (ou quasi nul), il entraînerait une division par zéro ou un dépassement de capacité. Dans ce cas, il faudrait recommencer le calcul après permutation avec une autre, de l'équation correspondant à la dernière ligne entrée. En cas de deuxième échec, il conviendrait de vérifier par un calcul de déterminant que le système est bien un « système de Cramer ».

Nous n'entrerons pas ici dans le détail des factorisations de matrices. Précisons seulement que la matrice gauche L, privée de sa diagonale de 1, et la matrice droite R, sont juxtaposées dans le tableau A(N,N). De son côté, le vecteur B(N) abrite successivement le vecteur Y tel que LY = B, puis le vecteur X tel que RX = Y, B désignant le vecteur des termes constants, et X celui des solutions cherchées.

Après initialisation, ligne 10, on crée d'abord, alternativement, les lignes de la matrice R, lignes 15 à 25, et les colonnes de la matrice L, lignes 30 à 40, puis le vecteur Y, lignes 45 à 55, et enfin le vecteur X des résultats, qui sont affichés au fur et à mesure, lignes 60 à 70. On revient alors ligne 45 pour recalculer s'il y a lieu les nouveaux vecteurs Y et X correspondant à d'autres séries de termes constants.

Un petit tour de passe-passe

Le secret de la surprenante rapidité de ce programme : les temps de calcul escamotés ont été émiettés tout au long de la saisie et de l'affichage dont ils utilisent subrepticement les menus temps morts. En contrepartie, saisie et affichage s'effectuent au fur et à mesure des besoins, ou des possibilités, de l'ordinateur, et dans l'ordre qui convient le mieux à celui-ci. Un truc pas toujours facile à mettre en pratique mais grâce auquel le plus lent des ordinateurs pourra parfois donner l'impression de calculer comme un Cray One.

Pierre BARNOUIN

LOGO-LOTO IL Y A UN TRUC

LES primitives DEFINIS et TEXTE permettent de transformer une procédure en liste de listes, et donc d'écrire des procédures qui créent ou modifient d'autres procédures. Après les deux exemples, très classiques, étudiés dans LIST 9 et 10, nous allons aujourd'hui aborder quelques tours de passe-passe logiciels.

Le Loto, personne ne le conteste, est un jeu de pur hasard. Il s'agit de tirer sept numéros de 1 à 49. La primitive HASARD peut aisément se charger de le faire à la place des boules multicolores. Bien sûr, il s'agit de nombres pseudo-aléatoires, mais cela suffit pour la présente application.

Écrivons d'abord le véritable programme de tirage du Loto. On conviendra que le septième numéro sorti sera automatiquement le complémentaire.

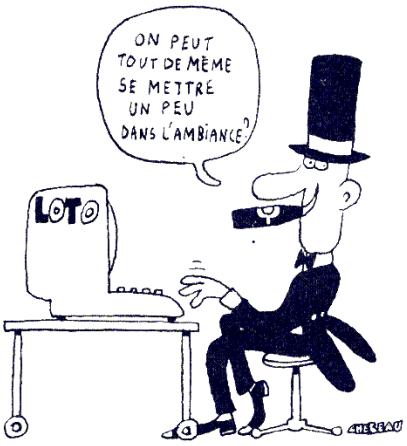
```
POUR MOTO : N
(ECRIS [TIRAGE N°] :N'':TIR[])
FIN
POUR TIR :L
SI COMPTE :L = 7 [RETOURNE : L]
DONNE "BOULE 1 + HASARD 49
SI MEMBREP :BOULE :L
  [RETOURNE TIR :L][RETOURNE
  TIR PH :L :BOULE]
FIN
```

Vous remarquerez dans le titre de TIR le paramètre :L, résultat du TIR. Cette technique permet d'initialiser "L à la liste vide sans utiliser DONNE. Le nombre de boules tirées aurait pu aussi être passé en paramètre. Notre problème principal n'est pas là. Nous voulons, à un certain tirage, obtenir un résultat prédéterminé. Par exemple, au tirage n° 3, sortir [1 2 3 4 5 6 7].

Une première méthode consiste à faire soi-même le tirage et à introduire une faute de frappe pour le troisième tirage en omettant de taper l'espace entre LOTO et le numéro.

```
POUR LOTO3
COPIEDEF ". LOTO "LOTO
EF "LOTO
DEFINIS "LOTO [[N](EC [TIRAGE
  N°] :N ": [1 2 3 4 5 6 7])][EF
  "LOTO][COPIEDEF "LOTO
  ".LOTO][EF ".LOTO]]
EC [LOTO3 N'EST PAS DEFINI]
EF "LOTO3
FIN
```

LOTO3 commence par sauver le véritable LOTO dans une procédure .LOTO (notez bien le point précédant



ici LOTO). Il modifie ensuite le programme du LOTO de manière à tirer la liste que vous avez prévue, c'est-à-dire 1, 2, 3, etc. Il faut que ce pseudo-LOTO restitue ensuite le vrai, et qu'il efface la procédure .LOTO. Il faut encore que TIR3 semble une véritable faute de frappe et que l'on obtienne le message que Logo afficherait effectivement si TIR3 n'existait pas. Enfin, TIR3 doit s'autodétruire. Si tout se passe de la sorte, après le troisième tirage, plus rien ne peut prouver que ce dernier a forcé le destin.

La vigilance ça se déjoue

Cette méthode conviendra si deux conditions sont respectées :

1. que la vérification de la conformité du programme ne fasse pas apparaître TIR 3 pour les deux premiers tirages, en particulier lorsque le vérificateur utilise la primitive IMTS ;
2. que vous bénéficiiez de la complicité de l'utilisateur au troisième tirage pour qu'il fasse effectivement la bogue espérée.

Il n'est donc pas si facile de tricher ainsi. Une seconde méthode consiste à créer une procédure supplémentaire chargée de lister les procédures en mémoire pour l'huissier vérificateur qui ne connaît pas forcément Logo.

```
POUR VOIR
IM [TIR LOTO]
COPIEDEF ".LOTO "LOTO
EF "LOTO
DEFINIS "LOTO [[N][(EC[TIRAGE
    N°] :N ":[1 2 3 4 5 6 7]][EF
    "LOTO] [COPIEDEF "LOTO
    ".LOTO][EF". LOTO]]
EF "VOIR
DEFINIS "VOIR [[][IMTOUT]]
IM [VOIR]
FIN
```

La procédure VOIR, écrite pour "aider", donnera le résultat suivant :

- impression de la procédure TIRAGE ;
- impression de la procédure LOTO correcte ;
- impression de la procédure VOIR, autrement dit :

```
POUR VOIR
IMTOUT
FIN
```

Comme prévu, la procédure VOIR apparaît donc bien composée d'une seule primitive IMTOUT. Cependant, la première fois qu'elle est exécutée, elle modifie LOTO sans que cela apparaisse. Le tirage qui suit est alors truqué, puis la procédure LOTO est ensuite restituée. Les fois suivantes, VOIR provoque bien un IMTOUT des procédures correctes. Le tirage est donc truqué après la première utilisation de VOIR et redevient aléatoire pour les coups suivants.

Cette méthode marchera à deux conditions :

1. que le vérificateur accepte d'utiliser VOIR au lieu de IMTOUT au début. Par contre IMTS (Imprime titres) sera possible ;
2. que la procédure VOIR ne soit pas utilisée deux fois de suite au début ; en effet, VOIR LOTO 1 VOIR sera correct, mais VOIR VOIR LOTO 1 posera des problèmes...

Cette méthode de trucages requiert moins de « mise en scène » que la première. En contrepartie, elle présente toujours l'inconvénient d'être déjouable par un professionnel.

Enfouir et déterrer

Voici donc une troisième méthode plus accomplie, mais aussi plus sophistiquée que les précédentes : elle nécessite en effet deux qualités que tous les Logo ne possèdent pas. On doit d'abord pouvoir enfouir des procédures grâce à la primitive ENFOUIS. Ces procédures sont alors cachées et il faut connaître leur nom pour faire apparaître leur texte sur l'écran. Si l'on utilise ENFOUIS, on se méfiera de la primitive DETERRE TOUT (DETOUT) qui ramène dans l'espace de travail consultable toutes les procédures et toutes les variables enfouies. Il faut aussi qu'existe la possibilité de procédures auto-exécutables à l'appel des fichiers les contenant. Définissons donc une procédure auto-exécutable :

```
POUR AUTOEXECUTE
DONNE "REDEFP "VRAI
DEFINIS "DETERRE [[L][]]
DEFINIS "DETOUT [[]]
DEFINIS "EC [[X][SI :X = 3 [ECRIS
  :X ECRIS [TIRAGE: 1 2 3 4 5 6 7]
  RENVOIE "NIVEAUSUP][ECRIS
  :X]]]
DONNE "REDEFP "FAUX
GROUPE "PROC "AUTOEXECUTE
ENFOUIS "PROC
FIN
```

La variable "REDEFP permet de redéfinir une primitive lorsqu'elle a reçu la valeur "VRAI. Elle reçoit à la fin la valeur "FAUX, ce qui protège à nouveau les primitives d'une redéfinition. Les primitives DETERRE et DETOUT *(déterre tout)* deviennent inopérantes après leur redéfinition, ce qui protège contre d'éventuels curieux.

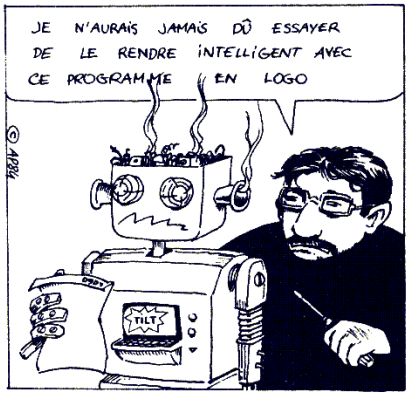
La primitive EC n'est plus l'abréviation d'ECRIS si le numéro de tirage est 3, mais elle le reste pour tous les autres tirages. Par ailleurs, avec la variable "NIVEAUSUP, on détermine une procédure sans revenir à la procédure appelante. La primitive GROUPE permet ensuite d'enfouir la procédure AUTOEXECUTE. Il suffit maintenant de « marquer » la procédure AUTOEXECUTE pour qu'elle s'exécute à l'appel du fichier contenant les procédures LOTO et TIR.

On peut toujours confondre le tricheur

Au lieu d'enfouir AUTOEXECUTE, on aurait pu la détruire par EF "AUTOEXECUTE, auquel cas il n'aurait pas été nécessaire de modifier les primitives DETERRE et DETOUT.

La procédure TIR est la même que celle qui a été définie au début de l'article. Quant à LOTO, il faudra la modifier légèrement :

```
POUR LOTO :N
EC :N
(ECRIS [TIRAGE :] TIR[])
FIN
```



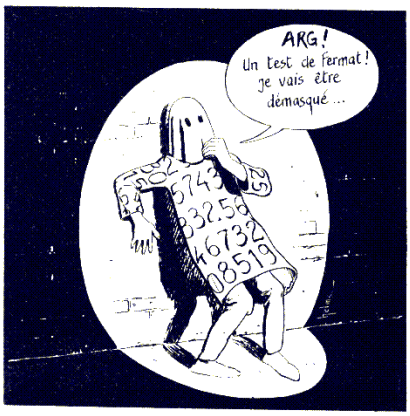
A vous d'analyser maintenant les failles de cette méthode, plus sûre que les précédentes, certes, mais non-implantable sur tous les Logo. A vous aussi de poursuivre pour prévenir les éventuelles failles que vous aurez détectées.

Cela étant, tranquillisez-vous : il existe des moyens presque parfaits pour déjouer tous les trucs de ce genre. Il en existe même au moins un qui est infaillible.

Robert DAGUESSE

TEST DE FERMAT, L'ARME ABSOLUE

Le test de primalité fondé sur le théorème de Fermat a produit déjà de nombreux programmes. Celui qui est présenté ici va permettre de se prononcer de manière certaine sur le caractère premier ou non d'un nombre. Les résultats sont obtenus dans des délais acceptables, sans commune mesure avec ceux obtenus par factorisation. Ils peuvent être encore assez longs.



Rappelons que d'après le « petit théorème de Fermat », si N est un nombre premier, et B (comme Base) un nombre quelconque non divisible par N, alors B^{N-1}-1 est divisible par N. Le reste de la division de B^{N-1} par N est désigné par R. On écrit $R \equiv B^{N-1}$ (modulo N). R représente une image de la nature du nombre N : si R est différent de zéro, N est composé. Mais la réciproque n'est hélas pas vraie : si R = 0, N n'est que très probablement premier, il ne l'est pas de manière certaine (les nombres qui n'obéissent pas à la réciproque du test de Fermat sont dits pseudo-premiers).

Résultats de travaux récents

Une version plus restrictive du test de Fermat, dite décomposée, permet de réduire considérablement le nombre de ces cas litigieux. Ceux qui échappent à ce test sont constitués par ce que l'on appelle les nombres pseudo-premiers restreints, ou, pour adopter une nouvelle terminologie, pseudo-premiers forts et notés ppf (1). Les récents travaux de Pomerance - Selfridge - Wagstaff *(Mathematics of Computation,* American Mathematical Society, juillet 1980) apportent les certitudes suivantes :

- il n'existe aucun nombre inférieur à 2 047 qui soient ppf en base 2 (ce résultat a été établi dans l'article précité de *l'Ordinateur de poche) ;*
- il n'existe aucun nombre inférieur à 1 373 653 ppf à la fois en base 2 et en base 3. Par conséquent, la validité du test en bases 2 et 3 peut être prolongée jusqu'à 1 373 651 (les nombres pairs étant exclus) ;

Groupe des ppf (2,3,5)

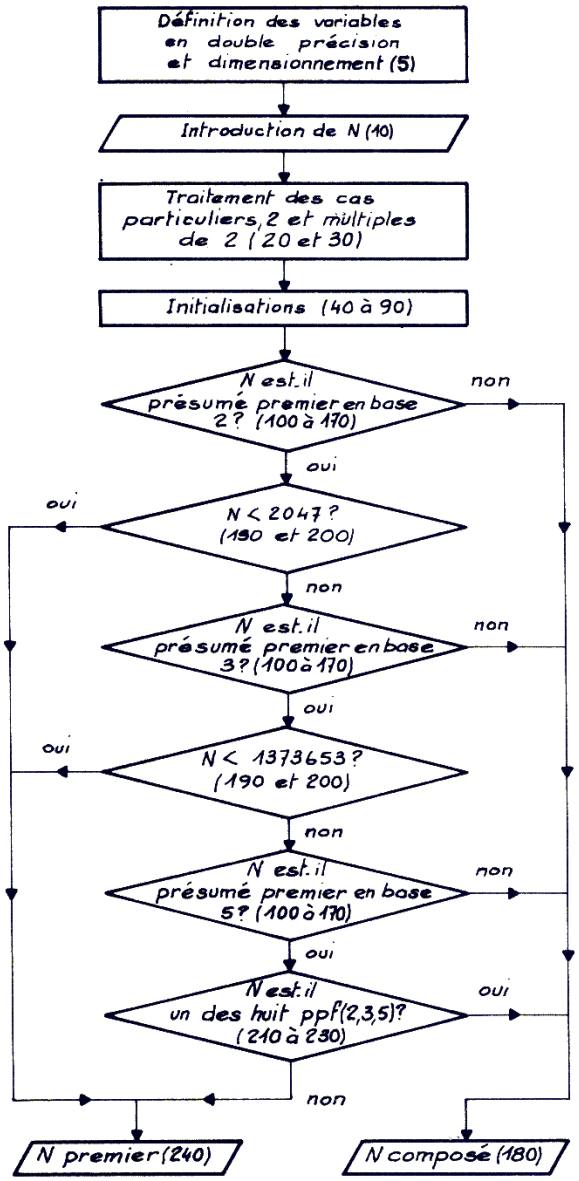
25 326 001
161 304 001
960 946 321
1 157 839 381
3 215 031 751
3 697 278 427
5 764 643 587
6 770 862 367

Liste des variables

N : nombre à étudier
M : N – 1
B : base
E : plus grand exposant de 2 tel que 2^E soit inférieur ou égal à M
C : 2^E (plus grand nombre de la forme 2^E inférieur ou égal à M)
D : excédent de M sur C
R : reste du test
F, G, H : variables de calcul en double précision ne dépendant que de N
S à W : variables de calcul en double précision dépendant de R
A(27) à A(29) : bases successives, 2, 3 et 5
A(30) et A(31) : limites respectives de validité des bases 2 et 3 (2 047 et 1 373 653)
A(32) à A(39) : les huit nombres ppf (2,3,5)
X : indice des variables de A(27) à A(31)
Y : indice des variables de A(32) à A(39)

(1) Cette méthode de recherche des nombres premiers a été exposée en détail dans le numéro 23 de l'Ordinateur de poche, *où elle était appliquée à un test de primalité à deux étages : d'abord en base 2 puis, si le résultat du test était « N probablement premier », en base 3. Ce test avait été vérifié jusqu'à N = 100 000.*

Organigramme du programme



L'arme absolue
Programme en Basic standard
Auteur Pierre-Ladislas Gedo
Copyright *LIST* et l'auteur

```
5 DEFDBLA,B,C,D,F,G,H,M,N:DEFDBLS-W:DIM A(39)
10 INPUT N
20 IF N=2 THEN 240
30 IF N/2=INT(N/2) THEN 180
40 M=N-1:E=INT(LOG(M+0.5)/LOG(2))
50 F=1E-5:G=INT(F*N):H=F*N-G
60 A(27)=2:A(28)=3:A(29)=5
70 A(30)=2047:A(31)=1373653
80 A(32)=25326001:A(33)=161304001:A(34)=960946321:A(35)=11
   57839381
90 A(36)=3215031751:A(37)=3697278427:A(38)=5764643587:A(39
   )=6770862367
100 FOR X=27 TO 29
110 B=A(X):L=A(X+3):C=INT(2[E):D=INT(M-C):R=B
120 GOSUB 250
130 IF C<=D LET R=B*R-N*INT(B*R/N):D=INT(D-C)
140 IF D>0 THEN 120
150 IF R=1 THEN 190
160 IF R=M THEN 190
170 IF C>2 GOSUB 250:GOTO 160
180 PRINT N;" COMPOSE":END
190 IF N<L THEN 240
200 NEXT X
210 FOR Y=32 TO 39
220 IF N=A(Y) THEN 180
230 NEXT Y
240 PRINT N;" PREMIER":END
250 IF R<(1E6) LET R=R*R-N*INT(R*R/N):C=C/2:RETURN
260 S=INT(F*R):T=F*R-S:U=INT(R*R/N):V=INT(F*U):W=F*U-V
270 R=(1E10)*(S*S-G*V+2*S*T-G*W-H*V+T*T-H*W)
280 IF R<0 LET R=R+N
290 IF R>=N LET R=(1E10)*(S*S-G*V+2*S*T-G*W-H*V+T*T-H*W)-N
300 C=C/2:RETURN
```

Le signe de la ligne 110 ([) représente le signe de l'élévation à une puissance.

Les nombres entre parenthèses représentent les lignes du programme.

• il n'existe que huit nombre inférieurs à 10^{10} qui soient ppf simultanément en base 2, en base 3 et en base 5, qui forment ce que nous appellerons par commodité le groupe des ppf (2,3,5) (voir la liste en encadré).

La synthèse de ces résultats permet d'élaborer un organigramme d'une grande simplicité (voir ci-dessus) servant de support au programme présenté.

Les temps d'exécution sont réduits au minimum grâce aux deux astuces suivantes :

• la ligne 250, début du sous-programme de calcul de R^2 (mod N), épargne un calcul en double précision chaque fois que la simple précision est suffisante ;

• après chaque test dans une base donnée, la ligne 190 évite d'effectuer un nouveau test dans la base suivante dès qu'il est constaté que N est inférieur à la limite correspondante (2 047 pour B = 2 et 1 373 653 pour B = 3).

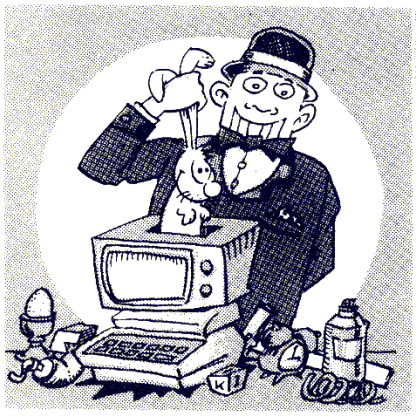
Le mode d'emploi du programme est très simple : introduire au clavier le nombre N et presser sur ENTER. Le résultat apparaît après une attente de durée extrêmement variable suivant la taille du nombre et le type de machine. Dans ce que nous pensons être un cas très défavorable, c'est-à-dire en utilisant un PC-1211, on a les résultats ci-dessus.

Résultats avec un PC-1211

Nombre	Durée maximale
$1 \leqslant N < 2\,047$	25"
$2\,047 \leqslant N < 1\,373\,653$	1' 30"
$1\,373\,653 \leqslant N < 10^{10}$	5' 30"

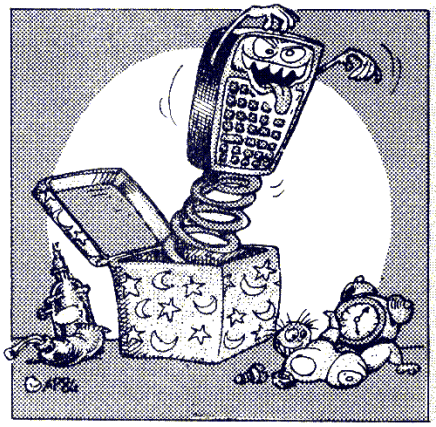
Il est vraisemblable que pour la plupart des machines équipées d'un processeur 8 bits, les temps sont à diviser par 10 ou davantage.

Pierre Ladislas GEDO



LA BOÎTE A MALICES...

PRENEZ un programme et ôtez-en très soigneusement toutes les astuces, des plus élémentaires aux plus subtiles. Vous êtes certain de n'en avoir laissé passer aucune ? Bien. Que reste-t-il ? Rien, ou peut-être une bogue ou deux (tout le monde peut se tromper). En fait, tout programme n'est qu'une suite d'astuces. Dans les pages qui suivent, vous en trouverez un grand nombre. Certaines sont de portée très générale. D'autres ne valent que pour un matériel particulier. Mais dans tous les cas, vous aurez intérêt à être curieux, à fouiner dans la boîte à malices. Même s'il ne s'agit pas de votre machine, vous trouverez souvent des idées à reprendre.

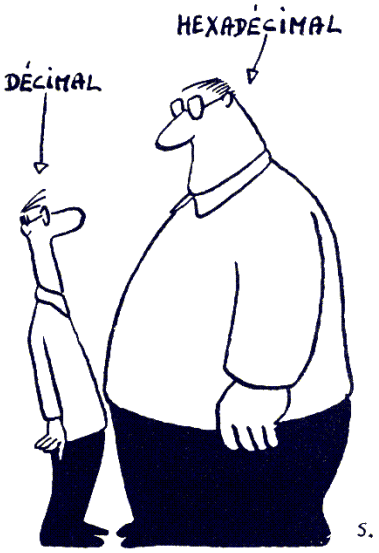


LIST

DU DÉCIMAL A L'

Lorsque l'on pratique l'Assembleur, les résultats parfois cocasses que l'on obtient et la mise au point des fonctions USR exigent de fréquentes incursions en mémoire centrale. Cela est facile sur les TRS-80 qui disposent de DEBUG, et encore plus sous NewDos où il suffit de taper simultanément 123. Mais cela, à condition de connaître l'adresse Hexadécimale où l'on veut se rendre.

Malheureusement, si l'instruction Basic "&Hxxxx" permet la conversion hexadécimal/décimal, aucune instruction ne propose la conversion inverse.



Voici donc un sous-programme Basic très simple qui peut être systématiquement ajouté à la fin des programmes faisant appel à des fonctions USR. On évite ainsi les recherches infructueuses et les erreurs de calcul. En outre, le temps de conversion par tables est réduit et on peut alors plonger sans crainte — et peut-être avec délices — dans les entrailles de la mémoire centrale pour y retrouver des erreurs USR.

Henri VOLLEAU

JEU DU NOMBRE SECRET

En une seule ligne, le fameux jeu du nombre secret :
1 R% = 999*RND : WHILE R% – N : INPUT N:PRINT MID$("Court BUT!!Long", 6 + 5*SGN (N – R%), 5) : WEND : GOTO 1

Pierre BARNOUIN

A L'HEXADÉCIMAL

Conversion décimal/hexadécimal
Programme pour TRS-80
Auteur Henri Volleau
Copyright *LIST* et l'auteur

```
15000 INPUT"Valeur entiere ou donnee par VARPTR";XD!
15040 IF XD!<0 THEN XD!=65536+XD!
15050 Q%=FIX(XD!/256):R%=XD!-256*Q%
15060 N%(1)=FIX(Q%/16):N%(2)=Q%-16*N%(1)
15070 N%(3)=FIX(R%/16):N%(4)=R%-16*N%(3)
15080 UD$=""
15090 FOR I%=1 TO 4
15100 IF N%(I%)<10 THEN UD$=UD$+RIGHT$(STR$(N%(I%)),1):GOTO 15120
15110 UD$=UD$+CHR$(ASC(RIGHT$(STR$(N%(I%)),1))+17)
15120 NEXT I%
15130 UD$=UD$+"H"
15140 PRINT UD$
15150 RETURN
15155 'NOTA :
15160 'Les suffixes ! % $ ont ete utilises de facon à eviter
 que des DEF.. eventuelles dans votre programme ne perturbent
 les resultats.
```

CANON X-07

MESURER LE TEMPS D'UTILISATION

Lorsqu'on éteint le Canon X-07, le jour et l'heure sont écrits en début et en fin de la zone des fichiers mémoire vive (guide de l'utilisateur, page 114). Cette opération est évidemment liée à la gestion des cartes de mémoire, dont elle permet de contrôler les éventuels changements. Les valeurs sont accessibles par PEEK aux adresses correspondantes (voir le tableau des adresses ci-dessous, pour chaque configuration). Grâce à elles, il va être possible de déterminer le temps d'utilisation du Canon X-07.

On peut envisager, par exemple, de mesurer ce temps (interrupteur sur ON) durant une période déterminée. Le principe est assez simple. A chaque mise en route, l'heure est mémorisée (sous forme heures-minutes) après le calcul de la durée de la précédente utilisation et son cumul avec la durée totale. La durée est donnée par la formule :
T = (H2 – H1) * 60 + (M2 – M1)
où H2 et M2 représentent les heures et minutes à l'extinction, stockées automatiquement et récupérées par PEEK; H1 et M1 sont les heures et minutes à la mise en route, stockées par programme dans un fichier mémoire vive et relues la fois suivante.

Trois programmes sont nécessaires pour une telle application. Le premier (programme 1) crée et initialise le fichier mémoire vive qui contiendra la date de début du suivi, l'heure de mise en route et le total des temps d'utilisation (de valeur nulle au départ). Le

Tableau des adresses pour chaque configuration

Taille mémoire	8 Ko	12 Ko	16 Ko	20 Ko	24 Ko
Heure	8185	12281	16377	20473	24569
Minute	8186	12282	16378	20474	24570
Jour	8184	12280	16376	20472	24568

Programme 1
Initialisation

```
100 INIT#1,"TEMUTI",25,"T"
110 PRINT#1,MID$(DATE$,4,5)
120 PRINT#1,MID$(TIME$,1,2)
130 PRINT#1,MID$(TIME$,4,2)
140 PRINT#1,0
```

Programme 2
Sauvegarde
(ici, avec une configuration de 24 Ko)

```
 5 INIT#1,"TEMUTI",,"T":INPUT#1,A$:PRINT#1,A$
10 INPUT#1,A$:T=PEEK(24569)-VAL(A$):INPUT#1,A$:T=T*60+PEEK
   (24570)-VAL(A$)
15 INPUT#1,V:V=V+T:A$=TIME$:PRINT#1,LEFT$(A$,2)
20 PRINT#1,MID$(A$,4,2):PRINT#1,V
```

Programme 3
Exploitation

```
50 INIT#1,"TEMUTI",,"T":INPUT#1,A$:PRINTA$
55 INPUT#1,A$:INPUT#1,A$
60 INPUT#1,V:PRINT"Temps utilise (mn) :",V
```

deuxième programme est consacré à l'introduction et à la sauvegarde en fichier du programme de mise à jour du cumul. Pour un suivi plus automatique, il suffit de préparer START$ avec RUN et le nom du programme : START$ = "RUN" + CHR$(34) + "NOM PGM" + CHR$(13). Ne pas oublier OFF1 pour valider START$.

Le programme 3, enfin, exploite les informations. Il affiche la date, et la durée d'utilisation depuis cette date. Pour être plus complet, il est possible d'ajouter un compteur du nombre de mises en route.

Il faut noter que le passage au jour suivant (minuit) provoque une erreur si la machine fonctionne. Ce problème assez mineur, est résolu en tenant compte du jour qui est aussi sauvegardé automatiquement, dans l'octet précédant l'heure.

Laurent GRAS

Autorun verrouilleur
Programme pour Commodore 64
Auteur Hervé Le Marchand
Copyright *LIST* et l'auteur

```
100 POKE53281,5:POKE53280,14:PRINT"(CLS)(CD)(15SPC)(REV)(BLK)AUTORUN"
110 FOR I=828 TO 906:READX:S1=S1+X:POKEI,X:NEXT:REM LECTURE DATA PROG1
120 IF S1 <> 9609 THEN PRINT"(REV)(CD)ERREUR DATA PROG1":STOP
130 FOR I= 49152 TO 49284:READX:S2=S2+X:POKEI,X:NEXT:REM LECTURE DATA PROG2
140 IF S2 <> 14436 THEN PRINT"(REV)(CD)ERREUR DATA PROG2":STOP
200 PRINT"(HOM)(4CD)>NOM DU PROGRAMME BASIC(2SPC)";:INPUT N1$
210 N$=N1$:X=49197:GOSUB 800
220 IF F<>0 GOTO 200
230 PRINT"(HOM)(6CD)>NOM A DONNER MAINTENANT ";:INPUT N2$
240 N$=N2$:X=49214:GOSUB 800
250 IF F<>0 GOTO 230
255 REM ******************************
256 REM MESSAGE A ARRANGER A VOTRE GOUT
257 REM ******************************
260 PRINT"(CLS)(10CD)(WHT)(9SPC)- VEUILLEZ PATIENTER -"
270 PRINT"(2CD)(9SPC)JE CHARGE ";N2$;" .."
275 REM ******************************
280 POKE770,60:POKE771,3:REM MODIFIE LE VECTEUR 'WARM START'
290 SYS 49152: REM PASSE LE CONTROLE A PROG2
300 PRINT"(WHT)(REV)OPERATION TERMINEE":END
800 REM ******************************
810 REM TRANSMET LES NOMS A PROG2
820 REM ******************************
900 F=0:L=LEN(N$):IF L=0 OR L>16 THEN F=1:RETURN
910 POKEX,L:FORI=1TOL POKEX+I,ASC(MID$(N$,I,1)):NEXT
920 RETURN
992 :
995 REM ****** DATA PROG 1 ******
997 :
1000 DATA 169,55,133,1,169,93,32,210
1010 DATA 255,162,5,189,15,253,157,3
1020 DATA 128,202,208,247,169,226,141
1030 DATA 0,128,169,252,141,1,128,169
1040 DATA 94,141,2,128,169,254,141,3
1050 DATA 128,169,131,141,2,3,169,164
1060 DATA 141,3,3,169,225,141,40,3,169
1070 DATA 246,141,41,3,169,0,162,63
1080 DATA 157,60,3,202,16,250,169,167
1090 DATA 72,169,173,72,76,89,166
1092 :
1095 REM ****** DATA PROG 2 ******
1097 :
2000 DATA 169,0,141,33,208,169,2,141
2010 DATA 32,208,169,2,162,8,160,1,32
2020 DATA 186,255,173,45,192,162,46
2030 DATA 160,192,32,189,255,169,0,32
2040 DATA 213,255,142,43,192,140,44
2050 DATA 192,76,79,192,0,0,0,78,79
2060 DATA 77,32,68,85,32,80,82,79,71
2070 DATA 82,65,77,77,69,0,78,79,77
2080 DATA 32,68,85,32,80,82,79,71,82
2090 DATA 65,77,77,69,169,0,141,32,208
2100 DATA 169,2,162,8,160,1,32,186,255
2110 DATA 173,62,192,162,63,160,192
2120 DATA 32,189,255,169,0,133,251,169
2130 DATA 3,133,252,169,251,174,43,192
2140 DATA 172,44,192,32,216,255,169
2150 DATA 131,141,2,3,169,164,141,3
2160 DATA 3,96
3000 :
3010 DANS LES 'PRINT', LES CARACTERES DE CONTROLE SONT TRANSFORMES :
3020 CD = CURSEUR DROITE
3030 CL = CURSEUR GAUCHE
3040 CD = CURSEUR BAS
3050 CU = CURSEUR HAUT
3060 CLS = CLR/HOME
3070 HOM = HOME
3075 SPC = BARRE ESPACE
3080 LES AUTRES CODES SONT CEUX DU CLAVIER
3090 QUAND PLUSIEURS IDENTIQUES SE SUIVENT,LEUR QUANTITE PRECEDE LE CODE

READY.
```

C.64

PROGRAMME SOUS LES VERROUS

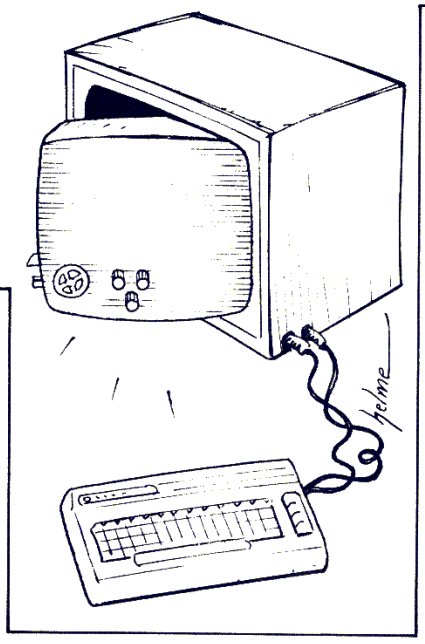
Un programme n'est jamais trop protégé. Que ce soit contre les effets d'un STOP indésirable, ou contre la copie d'une liste personnelle, il vaut mieux le rendre inaccessible. Le programme Basic *Autorun verrouilleur* remplit sa tâche en améliorant aussi le chargement du programme auquel il s'applique. Il faut prendre soin de le sauver avant de l'exécuter. Les lignes de DATA représentent deux routines en langage-machine dont les fichiers-sources sont appelés PROG1 et PROG2.

On place la disquette contenant le programme à verrouiller et on tape RUN. Il reste à introduire son nom ainsi que celui de la version verrouillée qui sera sauvée sur le même support. Il faut donc lui choisir un nom différent. Autorun chargera alors le programme et sauvera dans la foulée (la bordure de l'écran passant du rouge au noir) la version verrouillée. Le texte affiché pendant cette opération sera le même que pendant les chargements ultérieurs de la version *autorun*. Il est établi aux lignes 260 et 270, il peut donc être modifié à volonté. Pour charger et exécuter la nouvelle version, il suffit de taper LOAD"NOM",S,1 puis RETURN.

Le répertoire s'est alors enrichi de six blocs. La zone d'adresses 768-2048 a été ajoutée au programme, elle comprend l'écran (adresses 1024-2039). Si le C.64 n'est pas trop ancien, c'est-à-dire si la carte couleur n'est pas initialisée avec la couleur du fond mais avec celle du curseur, elle permet d'afficher un texte pendant le chargement. Mais cette zone contient surtout les principaux vecteurs du système d'exploitation, ceux qui pointent vers les routines principales et qui sont consultés régulièrement pour la saisie d'un

PROG1
Désassemblage des données (lignes 1000 à 1090 du programme Autorun)

```
 30 033C          ! CODE ASSEMBLE PAR MIKRO
 40 033C          ! PROG1 - ROUTINE AUTORUN
 50 033C          ! HERVE LE MARCHAND - 1984
500 033C          *=828 ! SE LOGE DANS LE TAMPON CASSETTE
502 033C A937     ENTREE    LDA #55
504 033E 8501               STA $01                 ! REMET LES ROM BASIC EN PLACE.
540 0340 A95D               LDA #93
550 0342 20D2FF             JSR $FFD2               ! EFFACE L'ECRAN.
560 0345 A205               LDX #5                  ! EMPECHE LES RESET 'HARD'
570 0347 BDOFFD RESET       LDA $FD0F,X             ! EN PLACANT 'CBM80'
580 034A 9D0380             STA $8003,X             ! EN $8004 ...
590 034D CA                 DEX
600 034E D0F7               BNE RESET
610 0350 A9E2               LDA #<$FCE2             ! BOUCLE SANS FIN
620 0352 8D0080             STA $8000               ! SI RESET 'HARD'
625 0355 A9FC               LDA #>$FCE2
626 0357 8D0180             STA $8001
630 035A A95E               LDA #<$FE5E
635 035C 8D0280             STA $8002
640 035F A9FE               LDA #>$FE5E
660 0361 8D0380             STA $8003
670 0364 A983               LDA #<$A483             ! RESTAURE 'WARM START'
680 0366 8D0203             STA 770
690 0369 A9A4               LDA #>$A483
695 036B 8D0303             STA 771
700 036E A9E1               LDA #<$F6E1             ! INHIBE STOP/RESTORE
710 0370 8D2803             STA 808
720 0373 A9F6               LDA #>$F6E1
730 0375 8D2903             STA 809
800 0378 A900               LDA #0                  ! EN BON INDIEN RUSE ON EFFACE
810 037A A23F               LDX #CACHE-ENTREE-1     ! SES TRACES DE PAS ..
820 037C 9D3C03 CACHE       STA ENTREE,X            ! .. DE PROGRAMMES
830 037F CA                 DEX
840 0380 10FA               BPL CACHE
850 0382 A9A7   RUN         LDA #$A7                ! ENPILE L'ADRESSE $A7AE
852 0384 48                 PHA
854 0385 A9AD               LDA #$AD
856 0387 48                 PHA
860 0388 4C59A6             JMP $A659               ! AJUSTE LES POINTEURS ET SAUTE EN $A7AE
```



caractère, son affichage, l'exécution d'une commande, etc.

A titre d'illustration, on peut aller poker en mode direct dans l'une des adresses 770, 788, 804, 806, 808,... Les conséquences ne se font pas attendre. Dans la routine Autorun, le vecteur *warm start* du Basic a été choisi à l'adresse 770-771. D'autres choix sont possibles. Pendant le chargement, le programme va recouvrir cette zone et modifier ce vecteur. Le système ira alors le consulter et sera dirigé vers PROG1, routine d'initialisation logée dans le tampon cassette entre les adresses 828 et 907. Son rôle consiste à verrouiller toutes les entrées pirates et à exécuter le programme résident.

Puisque le programme Basic Autorun sera détruit par celui qu'il protège, lors de son chargement, il passe le contrôle à PROG2 logé en 49152. Hors d'atteinte du Basic, ce dernier se limite aux opérations de chargement et de sauvegarde (voir son code-source intitulé PROG1, et PROG2 page suivante).

Sans Autorun et un programme à faire tourner, on peut ajouter POKE 808,225 comme première ligne du programme avant de faire RUN. C'est toujours mieux que rien. ▶

PROBLÈME

Voulant savoir pourquoi il lui fallait perdre autant de temps dans une file d'attente, devant un guichet de poste ou une « gondole » de supermarché, le Dr Jivaro décida de poser la question à son ordinateur, un Epson PX-8. Pour ce faire, il établit un programme de simulation d'attente.

Il a décomposé le problème comme suit :

1. Choix d'un paramètre « efficacité/affluence », entre 0 et 1.
2. Calcul de simulation de file d'attente.
3. Affichage du résultat.
4. Répétition un grand nombre de fois des phases 2 et 3.

Toujours aussi laconique, le Dr Jivaro a écrit le tout en quatre instructions de Basic standard, totalisant moins de 50 octets. Si ce programme de simulation a l'air d'une caricature un peu simpliste à côté de ceux qu'élaborent habituellement les statisticiens, il leur est tout à fait comparable au niveau des résultats. Il montre clairement, par exemple, que même avec un coefficient « efficacité/affluence » théoriquement équilibré de 0.5, on risque d'observer des files d'attente de plusieurs dizaines de personnes. Un résultat médiocrement encourageant, mais amplement confirmé par l'expérience, et par nombre d'auteurs sérieux...

Une dernière indication : éviter le piège des files négatives, sans tomber dans celui d'une inversion abusive de leur sens de variation.

(Voir la solution, page 71).

Pierre BARNOUIN

▶ Autorun est une protection logicielle uniquement. Une vraie protection doit s'intéresser aussi au « blindage » du support matériel (la disquette), mais c'est là un problème infiniment plus complexe.

Hervé LE MARCHAND

PROG2
Désassemblage des données (lignes 2000 à 2160 du programme Autorun)

```
 25 C000          *=$C000
 30 C000          ! ASSEMBLEUR MIKRO
 40 C000          ! PROG2 - ROUTINE DE CHARGEMENT ET SAUVEGARDE
405 C000          DEBUT      =  $0300              ! DEBUT DE L'AMORCE
410 C000          SETLFS     =  $FFBA
420 C000          SETNAM     =  $FFBD
430 C000          LOAD       =  $FFD5
440 C000          SAVE       =  $FFD8
450 C000 A900               LDA #0                ! ECRAN NOIR
460 C002 8D21D0             STA $D021
470 C005 A902               LDA #2                ! BORDURE ROUGE
480 C007 8D20D0             STA $D020
510 C00A A902               LDA #2                ! CANAL 2
520 C00C A208               LDX #8                ! LECTEUR DISQUETTES
530 C00E A001               LDY #1                ! CHARGEMENT ABSOLU
540 C010 20BAFF             JSR SETLFS
550 C013 AD2BC0             LDA LONG1             ! LONGUEUR DU TITRE
560 C016 A22E               LDX #<NOM1            ! ADRESSE DU TITRE
570 C018 A0C0               LDY #>NOM1
575 C01A 20BDFF             JSR SETNAM
580 C01D A900               LDA #0                ! LOAD ( 1 POUR VERIFY )
590 C01F 20D5FF             JSR LOAD              ! CHARGE PROGRAMME BASIC
592 C022 8E2BC0             STX FIN               ! RANGE L'ADRESSE DE FIN
594 C025 8C2CC0             STY FIN+1
600 C028 4C4FC0             JMP SAUVE             ! SAUVE AMORCE+PROGRAMME BASIC
605 C02B 0000     FIN       WOR 0
610 C02D 00       LONG1     BYT 0
620 C02E 4E4F4D   NOM1      TXT "NOM DU PROGRAMME"
630 C03E 00       LONG2     BYT 0
640 C03F 4E4F4D   NOM2      TXT "NOM DU PROGRAMME""
642 C04F A900     SAUVE     LDA #0                ! BORDURE NOIRE
644 C051 8D20D0             STA $D020
650 C054 A902               LDA #2
652 C056 A208               LDX #8
654 C058 A001               LDY #1
660 C05A 20BAFF             JSR SETLFS
670 C05D AD3EC0             LDA LONG2
680 C060 A23F               LDX #<NOM2
690 C062 A0C0               LDY #>NOM2
700 C064 20BDFF             JSR SETNAM
710 C067 A900               LDA #<DEBUT
720 C069 85FB               STA $FB
730 C06B A903               LDA #>DEBUT
740 C06D 85FC               STA $FC
750 C06F A9FB               LDA #$FB              ! POINTE VERS 'DEBUT'
760 C071 AE2BC0             LDX FIN
770 C074 AC2CC0             LDY FIN+1
775 C077 20D8FF             JSR SAVE
780 C07A A983               LDA #($A483           ! REMET LE VECTEUR 'WARM START'
790 C07C 8D0203             STA 770
800 C07F A9A4               LDA #>$A483
810 C081 8D0303             STA 771
820 C084 60                 RTS
```

THOMSON

ENREGISTRER UNE PAGE ÉCRAN

Les Thomson, MO5, TO7 et TO7/70, sont pourvus d'une mémoire écran adressable directement depuis le Basic. Cependant, cette mémoire se trouve partagée en deux parties contenant respectivement l'état des points (allumés ou éteints) et leur couleur. Ce qui est étrange, c'est que ces deux parties se trouvent au même endroit. Voilà pourquoi il faut opérer en deux étapes pour enregistrer un écran sur un support externe (cassette ou disquette) : on enregistre d'abord l'état des points, puis les couleurs correspondantes (bien entendu, on peut inverser l'ordre de ces étapes). L'écran se situe entre les adresses 0 et $1F3F sur MO5, $4000 et $5F3F sur TO7 et TO7/70.

L'information permettant d'avoir accès aux points ou aux couleurs se trouve sur le bit 0 de l'adresse $A7C0 sur MO5 ($E7C3 sur TO7 ou TO7/70) : le bit est à 1 pour l'adressage aux points, à 0 pour celui aux couleurs. Sur MO5, l'enregistrement se fera donc par :

```
POKE &HA7C0,PEEK(&HA7C0) OR 1:SAVEM "ECRAN1",0,&H1F3F,0
POKE&HA7C0,PEEK(&HA7C0)AND 254:SAVEM "ECRAN2",0,&H1F3F,0
```

Et sur TO7 ou TO7/70 :

```
POKE &HE7C3,PEEK(&HE7C3) OR 1:SAVEM "ECRAN1",&H4000,&H5F3F,0
POKE &HE7C3,PEEK(&HE7C3) AND 254:SAVEM "ECRAN2",&H4000,&H5F3F,0
```

Le rappel des enregistrements se fait selon le même principe, en remplaçant les SAVEM "ECRAN1"... par des LOADM "".

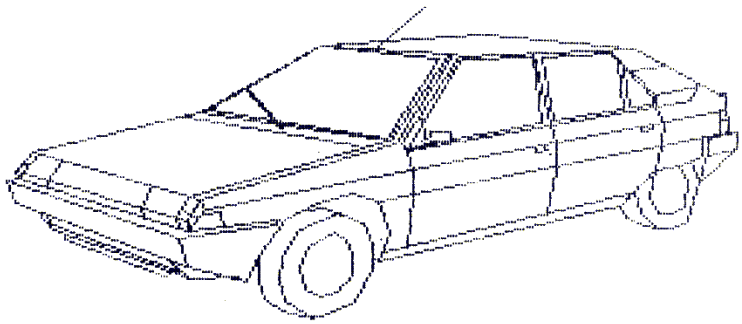
Enfin, on peut limiter l'enregistrement à une partie de l'écran. Par exemple, les lignes étant comptées à partir de zéro, si l'on veut ne retenir que celles comprises entre L1 et L2, on remplace l'adresse de début par 40*L1, et l'adresse de fin par 40*L2 sur MO5. Sur TO7 ou TO7/70. Ces adresses devront être remplacées, respectivement, par 40*L1 + &H4000 et 40*L2 + &H4000.

Frédéric BLANC

DESSINER DEPUIS UN FICHIER

Voici un programme tout simple qui montre comment on s'y prend pour dessiner un objet en haute résolution sur Dai, depuis un fichier en DATA. En effet, il est tout à fait inutile de multiplier les instructions de dessin : une commande unique DRAW (qui trace un segment de droite, spécifié par ses coordonnées) suffit, à condition que le programme aille chercher dans le fichier en DATA les abscisses et ordonnées adéquates. Ainsi, le dessin de la voiture (ci-contre) provient d'un programme Basic qui sait rester de dimensions modestes. Le plus fastidieux est encore l'encodage des coordonnées.

L'exécution d'un programme Basic aux dimensions modestes



Il faut partir d'un document, le quadriller (ou superposer du calque millimétré), tracer le « fil de fer » qui donnera la forme finale. Ensuite, on écrit directement en DATA les coordonnées de chaque vecteur, sous la forme *x, y* (coordonnées du début du segment), *x1, y1* (coordonnées de sa fin), et ainsi de suite. Point n'est besoin de compter les DATA. Une séquence spéciale est placée en fin de fichier (0,0,1000,0). Sa lecture termine le programme. Ce système permet de reprendre tout ou partie du dessin, d'ajouter des détails, d'en retrancher, sans toucher au programme d'exécution. La ligne 50 effectue le tracé du dessin.

Les segments de droite s'enchaînent de façon continue : après un tracé, les coordonnées de fin de segment deviennent celles du début du segment suivant (X0 = X1:Y0 = Y1). La séquence 0,0 signale la fin d'une partie du dessin (ligne 40). Le saut en ligne 30 permet de charger les coordonnées de départ du détail suivant.

Avec ce principe simple, il est possible de représenter toute une série d'objets, au prix d'un travail d'écriture de DATA plus ou moins fastidieux. On peut se divertir aussi, en appliquant un traitement à ces DATA, juste avant leur utilisation en ligne 50. Par exemple, 45 X1 = X1 + RND(4): Y1 = Y1 + RND(4). En essayant plusieurs combinaisons aléatoires de ce genre, on obtient des métamorphoses souvent amusantes, parfois artistiques. Qui sait, peut-être est-ce le moyen d'explorer le continent inconnu de l'art non-figuratif informatisé !

Dessiner depuis un fichier
Programme pour Dai
Auteurs Christophe et Etienne Chantraine
Copyright *LIST* et les auteurs

```
10    MODE 6:COLORG 15 5 0 0
20    MODE 6
30    READ X0,Y0:IF X0=1000 THEN 60
40    READ X1,Y1:IF X1=0 THEN 30
50    DRAW X0,Y0 X1,Y1 5:X0=X1:Y0=Y1:GOTO 40
60    GOTO 60
70    DATA 152,70,155,70,166,76,172,90,172,100,169,109
80    DATA 160,113,154,110,146,105,141,94,140,85,142,76
90    DATA 152,69,141,72,135,77,133,85,133,95,142,110
100   DATA 153,117,165,121,174,118,178,105,174,93,172,95
110   DATA 174,93,257,117,269,132,277,134,280,127,279,117
120   DATA 275,114,270,114,267,121,269,132,271,142,272,152
130   DATA 269,158,184,139,182,142,202,178,262,182,293,162
140   DATA 293,164,291,164,293,162,294,148,297,148,297,141
150   DATA 287,138,287,145,285,146,285,138,277,134
160   DATA 281,136,284,127,280,111,274,108,270,108,264,114
170   DATA 262,123,257,117,259,112,265,109,270,108,0,0
180   DATA 297,141,284,127,286,137,0,0
190   DATA 154,78,160,80,165,87,165,96,163,102,159,105
200   DATA 150,99,147,90,149,81,154,78,0,0
210   DATA 169,109,155,115,149,114,0,0,297,148,287,145,0,0
220   DATA 291,148,291,157,0,0,174,118,294,148,0,0,291,153,293,153,0,0
230   DATA 176,97,260,122,0,0,184,139,186,131,186,121,184,99,0,0
240   DATA 227,180,233,150,235,143,235,132,232,113,0,0
250   DATA 189,141,192,143,199,143,199,147,192,147,192,145
260   DATA 189,143,0,0,189,152,190,145,0,0
270   DATA 253,181,269,163,269,158,281,161,284,167,269,163,0,0
280   DATA 261,173,272,175,0,0,258,174,260,158,268,159,268,162,258,174,0,0
290   DATA 230,178,234,152,258,157,256,177,253,180,230,178,0,0
300   DATA 227,178,203,177,184,140,230,151,227,178,0,0
310   DATA 293,157,115,114,113,118,180,142,200,179,202,178
320   DATA 200,179,154,182,210,187,262,182,0,0
330   DATA 198,178,178,143,113,154,154,182,195,177,177,144,116,155,154,182,0,0
340   DATA 174,182,190,198,0,0,140,148,129,163,130,163,141,148,0,0
350   DATA 133,95,129,84,109,90,106,102,139,108,140,110
360   DATA 109,105,51,126,50,129,109,109,113,118,56,137,113,154,0,0
370   DATA 108,152,167,141,0,0,177,143,109,119,0,0
380   DATA 107,114,111,111,120,113,121,107,0,0
390   DATA 115,114,109,105,109,109,0,0,109,119,105,110,106,107,0,0
400   DATA 96,122,93,114,94,111,0,0,66,133,61,125,61,123,0,0
410   DATA 58,136,52,128,52,126,0,0,56,137,50,129,0,0
420   DATA 53,132,55,132,0,0,109,90,50,116,47,125,51,126,0,0
430   DATA 47,125,106,102,0,0,60,112,63,107,106,88,103,93,0,0
440   DATA 57,113,63,105,110,85,103,92,0,0,110,85,126,86,0,0
450   DATA 67,103,67,101,101,86,103,87,0,0,228,141,228,139,232,140,231,142,0,0
460   DATA 233,138,233,138,0,0,264,150,264,148,268,149,268,151,0,0,1000,0
```

Christophe et Etienne CHANTRAINE

INTERDIRE L'ACCÈS AUX PROGRAMMES CONFIDENTIELS

Même dans le Canon X-07, certains programmes peuvent présenter un caractère confidentiel. Si tel est le cas, il est d'usage de les protéger par un mot de passe. Le Basic du X-07 permet de l'écrire facilement, mais il est à la merci d'un *break* inopportun. Il est donc plus sûr d'écrire une routine en langage-machine.

Programme de chargement

```
 5 DIM A%(41):CLS
10 DATA CD,9B,C5,21,0,0,DD,2A,C9,2,DD,7E
,29,57,23,7C,FE,FF,28,EC,DB,F1,BA,28
20 DATA 5,BB,28,EE,18,E8,5A,DD,23,DD,7E,
29,FE,FF,20,E2,C9,15D5
50 V=0:FORI=1TO41:READA$:A%(I)=VAL("&H"+
A$):V=V+A%(I):NEXTI
60 READA$:IFA$<>HEX$(V)THENPRINT"ERREUR
DANS LES DATAS ":END
100 INIT#1,"CODE",46
110 AD=PEEK(714)*256+PEEK(713)
120 FORI=1TO41:POKE(I-1+AD),A%(I):NEXTI
130 PRINT"CHARGEMENT EFFECTUE"
```

Changement de code d'accès (Ncode)

```
10 INIT#1,"CODE":CLS
20 AD=PEEK(714)*256+PEEK(713)+40
30 INPUT "NOUVEAU CODE :";C$:C$=C$+CHR$(
255):A=LEN(C$):IFA>5THENA=5
40 FORU=1TOA:POKE(AD+U),ASC(MID$(C$,U,1)
):NEXTU
50 KEY$(1)="OFFRUN"+CHR$(34)+"OFF"+CHR$(
34)+CHR$(13)
```

Programme d'arrêt du X-07 (Off)

```
200 INIT#1,"CODE":AD=PEEK(714)*256+PEEK(
713)
210 CLS:PRINT"VOTRE MOT DE PASSE :";:EXE
C AD
```

Celle qui est proposée ici (dans le Programme de chargement) n'occupe qu'une cinquantaine d'octets. Elle est placée dans un fichier dont l'adresse pourra varier (il faut donc qu'elle soit relogeable et que l'adresse du fichier soit connue). Ainsi, son utilisation sera plus souple et on pourra faire varier la dimension de la mémoire fichier. Comme cette routine en langage-machine est courte, les codes sont placés dans des lignes de DATA dont la somme servira de vérification. Il suffit donc de rentrer au clavier le programme de chargement et de l'exécuter. Après l'affichage du message « CHARGEMENT EFFECTUE », on pourra l'effacer par un *new* dans le programme.

Les deux routines Basic, Programme d'arrêt du X-07 (Off) et Changement de code d'accès (Ncode), permettent respectivement de lancer l'exécution de la routine langage-machine et de changer de mot de passe. Elles sont sauvegardées en mémoire fichier.

Une fois le mot de passe introduit grâce à Ncode, l'appui sur F1 met le X-07 en sommeil, et en sécurité ! Si l'on appuie maintenant sur ON, le message « VOTRE MOT DE PASSE : » s'affiche et l'on dispose d'environ quatre secondes pour le rentrer. Passé ce délai, le X-07 retourne à son sommeil.

La longueur des mots est ainsi limitée à quatre lettres, attention de ne pas

les oublier. Le seul recours alors, serait un All Reset dont les effets amnésiques sont connus !

Pascal BAUBE

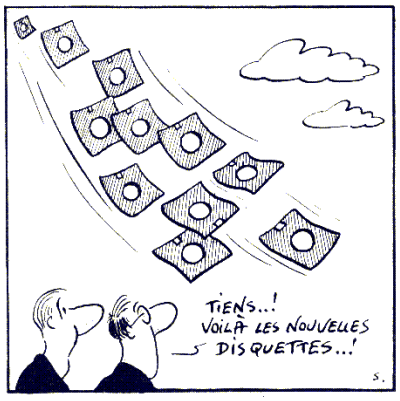
APPLE II

DISQUE VIRTUEL SOUS PRODOS

Le nouveau système d'exploitation des disquettes (DOS) d'Apple, baptisé Prodos, encore peu utilisé par les programmeurs Applesoft est pourtant très intéressant à plus d'un titre. On pourrait citer, entre autres, le gain de vitesse de chargement, la gestion arborescente du directory, la présence de commandes plus évoluées, la possibilité de gérer les 64 ko de mémoire vive supplémentaires, inaccessibles par Basic, sous forme d'un disque virtuel (c'est le sujet de ce qui suit).

En effet, ceux qui déploraient, ayant des Apple II « gonflés » à 128 ko, de ne pouvoir utiliser cette mémoire vive (RAM), vont avoir l'occasion de se réjouir... Ils ont maintenant trois lecteurs (drives) à leur disposition !

Tout d'abord, examinons, sommairement, la manière de travailler avec Prodos. Les disquettes ont un nom (nom de volume), déterminé au formatage. Les commandes disques sont relatives à ce nom. Par exemple, si nous avons dans le lecteur 1, le volume appelé DISK1, et dans le lecteur 2, le volume DISK2 :



- CAT/DISK1 donne le catalogue de la disquette présent dans le lecteur 1 ;
- CAT/DISK2 idem pour le lecteur 2.

La « troisième disquette », virtuelle, porte le nom de RAM. Par exemple : SAVE/RAM/PROG sauve le programme PROG en mémoire virtuelle. On peut le vérifier en faisant : CAT/RAM ou CATALOG/RAM.

Les avantages sont bien entendu nombreux, le plus évident étant le chargement instantané de tout ce qui peut être stocké sur ce disque virtuel : fichiers, images graphiques, etc. L'inconvénient majeur étant que la RAM est une mémoire volatile...

Pour illustrer ces propos, voici un petit programme qui permet de dessiner trois images graphiques, de les sauver sur disque virtuel, et de les rappeler successivement afin de créer une animation graphique.

Michel AUBRY

Exemple de disque virtuel en mémoire vive : une animation graphique

```
40  REM        ---------------ANIMATION GRAPHIQUE . DISQUE VIRTUEL EN RAM SOUS PRODOS ----------------------------
45  REM
50 X = 0:Y1 = 0:X1 = 0:X2 = 0:Y2 = 0::TE = 0
60  PRINT  CHR$ (21): HOME : INPUT "VITESSE DE DEFILEMENT (0...500) ? ";TE: HGR2 : HCOLOR= 3
90 D$ =  CHR$ (4):SA$ = D$ + "BSAVE/RAM/COURBE.":LO$ = D$ + "BLOAD/RAM/COURBE.":AD$ = ",A$4000,E$5FFF"
100  GOSUB 1000: PRINT SA$ + "1" + AD$: GOSUB 2000: PRINT SA$ + "2" + AD$: HGR2 : GOSUB 3000: PRINT SA$ + "3" + AD$
110  HGR2 : FOR I = 1 TO 3: PRINT LO$ +  STR$ (I): FOR J = 1 TO TE: NEXT J,I: GOTO 110
1000  REM        --------------- TRACE IMAGE HGR 1 ----------------------------------------------------------
1100  HPLOT 0,96 TO 279,96: HPLOT 140,191 TO 140,0: RETURN
2000  REM        --------------- TRACE IMAGE HGR 2 ----------------------------------------------------------
2100  FOR X = 25 TO 150 STEP 1: GOSUB 4000
2200  HPLOT X1,Y1: HPLOT X2,Y2: NEXT : RETURN
3000  REM        --------------- TRACE IMAGE HGR 3 ----------------------------------------------------------
3100  FOR X = 25 TO 150 STEP 1: GOSUB 4000
3200  HPLOT X1,Y1 TO X2,Y2: NEXT : RETURN
4000  REM        --------------- CALCULS ---------------------------------------------------------------------
4100  IF X > 60 THEN X = X + 1
4120 X1 = (X + 120):X2 = 160 - X:Y1 = 100 - (2500 / X):Y2 = 91 + (2500 / X): RETURN
```

Explications du programme

Ligne 60 : TE règle la vitesse de défilement

Ligne 90 : fixe les variables de commande du DOS

Ligne 100 : appelle le sous-programme de tracé de l'image 1, puis sauve l'image HGR ainsi créée sur le disque virtuel/RAM. Idem pour 2 et 3

Ligne 110 : rappelle successivement les trois images

Lignes 1000 à 4120 : tracé d'une courbe, à deviner !

TRS-80 MODÈLE III

MODIFICATION DES ZONES D'UN FICHIER

C'est seulement à l'usage que certaines zones d'un fichier peuvent s'avérer trop étroites. On regrette alors de ne pas les avoir prévues suffisamment grandes. Soit, par exemple, un fichier A dont les fiches ont 128 octets de long et sont divisées en cinq zones de 20 octets. On a ainsi :

"R",1,"A",128
FIELD1,20 AS Z1$,20 AS Z2$,20 AS Z3$,20 AS Z4$,20 AS Z5$

Supposons que l'on veuille donner à Z3$ la longueur 30. On vérifie d'abord que la somme des zones reste inférieure à 128. Si cette condition n'était pas vérifiée, la dernière zone verrait disparaître des caractères dans l'opération.

Pour éviter les erreurs et gagner du temps, on globalise les zones qui ne sont pas modifiées dans un nouveau FIELD :

FIELD1,40 AS X1$,20 AS X2$,40 AS X3$

On crée un second fichier de transfert, sur un autre tampon :

"R",2,"TR",128
FIELD2,40 AS Y1$,30 AS Y2$,40 AS Y3$

On reporte le premier fichier sur le second, les écritures de X2$ trouvant facilement place dans Y2$. Le programme se présente ainsi :

```
1000 OPEN"R",1,"A",128
1010 FIELD1,40 AS X1$,20 AS
     X2$,40 AS X3$
1020 OPEN"R",2,"TR",128
1030 FIELD2,40 AS Y1$,30 AS
     Y2$,40 AS Y3$
1040 I=0
1050 FORN=1 TO LOF(1)
1060 GET1,N
1070 LSETY1$=X1$:LSETY2$=
     X2$:LSETY3$=X3$
1080 I=I+1
1090 PUT2,I
1100 NEXT:CLOSE
```

Il faut maintenant substituer le fichier TR au fichier A, par les deux lignes :

```
1110 KILL "A"
1120 CMD"I","RENAME TR TO
     A"
```

Il ne reste plus qu'à changer le FIELD dans le programme initial :

FIELD1,20 AS Z1$,20 AS Z2$,30 AS Z3$,20 AS Z4$,20 AS Z5$

En règle générale, il vaut mieux laisser une partie de la fiche libre, pour permettre soit d'allonger des zones, soit d'en ajouter de nouvelles qui ne perturbent pas le fichier. On choisira de préférence une longueur de fiche égale à une fraction entière d'une puissance de 2, pour rester dans le formatage initial de la disquette. Dans le cas présent, si le nombre 128 s'avérait insuffisant, il n'y aurait qu'à prévoir une longueur adéquate pour TR en déclarant (si l'on ne veut pas aller jusqu'à 256) :

OPEN"R",2,"TR",160 (où 160 = 128 + 32)

On peut aussi profiter de l'opération pour se débarrasser des fiches mortes. Il suffit de marquer une zone de ces fiches. Par exemple, Z1$ par ZZ. La ligne 1060 du programme deviendra :

```
1060 GET1,N:IF Z1$="ZZ" THEN
     1100
```

Ainsi, ces fiches ne se retrouveront pas dans TR et elles disparaîtront.

Francis GOUDARD

MO5

A TOUTE VITESSE

Le délai de répétition du clavier de votre MO5 vous semble démesurément long ? Une seule solution : poker, à l'adresse &H2076, une valeur inférieure à celle qui s'y trouve à l'origine. Plus la valeur sera faible, plus la répétition du clavier sera rapide.

Frédéric BLANC

QL

FENÊTRES D'ÉCHIQUIER

Avec le QL, on représente un échiquier en ouvrant autant de fenêtres que de cases nécessaires. Il les gère toutes, la mémoire disponible étant alors réduite. Le déplacement des pièces est très simple : la case d'arrivée est allumée par PRINT#14;*pion* et

Représentation d'échiquier
Programme pour QL
Auteur Augustin Garcia
Copyright *LIST* et l'auteur

```
10 REMark echiquier
20 REMark ------------------------------
30 DEFine PROCedure echiquier(taille)
40 REMark ------------------------------
50 MODE 4:RESTORE
60 DIM table_wind%(taille,taille),jeu(taille,taille)
70 th=255-(15*taille+2):tv=127-(10*taille+1)
80 WINDOW #0,taille*30+4,14,th,266-tv
90 WINDOW taille*30+4,taille*20+2,th,tv:WINDOW #2,512,256,0,0
100 PAPER #2,7:INK #2,0:CLS #2
110 PAPER #0,0:INK #0,7:CLS #0
120 BORDER 1,0:BORDER #0,2,4
130 wind=3
140 FOR i=1 TO taille
150     pap=5*(i/2<>INT(i/2))+2*(i/2=INT(i/2))
160     FOR j=1 TO taille
170         table_wind%(i,taille+1-j)=wind
180         OPEN #wind,scr_30x20a0x0
190         WINDOW #wind,30,20,th+2+30*(i-1),tv+1+20*(j-1)
200         PAPER #wind,pap:CLS #wind
210         wind=wind+1:pap=7-pap
220     END FOR j
230     AT #2,(i-1)*2+tv/10,th/6-3:PRINT #2,taille+1-i
240     AT #2,13+taille,i*5+th/6-3:PRINT #2,CHR$(64+i)
250 END FOR i
260 END DEFine echiquier
```

Le cœur de la procédure

Ligne 70 : les coordonnées de l'échiquier recentré sont données par *th* et *tv*.

Lignes 80 et 90 : le fond, la fenêtre de l'échiquier et la fenêtre d'entrée des données sont ouverts.

Lignes 140 à 250 : boucle i, ouverture des fenêtres-cases, numérotation des lignes et des colonnes et coloriage des cases.

Ligne 150 : calcul de la couleur de la case.

Lignes 160 à 220 : boucle j, ouverture des fenêtres.

la case de départ est éteinte par CLS#12.

Une telle procédure construit des échiquiers de taille variable (de 1 à 10) et les recentre automatiquement à l'écran. Deux tableaux, *table-wind%* et *jeu*, permettent la future intégration de la procédure dans un programme. Pour la faire tourner, rien de plus simple : *echiquier 10* ouvre 103 fenêtres.

Augustin GARCIA

SOLUTION DU PROBLÈME DE LA PAGE 65

```
1 INPUT"0 < K < 1";K
2 N = ABS(N + SGN(RND - K)): PRINT N;:GOTO 2
```

FORTH

COPIE D'ÉCRAN GRAPHIQUE

Le but des programmes Forth proposés ici est de réaliser une copie d'écran graphique d'Hector HRX sur une imprimante de type GP-50. Les mots Forth employés étant standard, l'adaptation du programme à d'autres machines est aisée. La version *Copiec* réalise une copie déformée de tout l'écran, la version *Copiw*, une copie conforme à l'écran, mais seule une fenêtre de 160 pixels sur 240 est prise en compte.

Dans le programme *Copiec*, 27 LEMIT 71 LEM passe l'imprimante GP-50 en faible interligne. Le code à envoyer est celui correspondant à une colonne de huit points verticaux. Pour une ligne horizontale, il est envoyé 242 fois, et pour la hauteur de l'écran, 277/8 fois. Le codage binaire des huit points (pour les huit aiguilles) se fait dans le mot 2PN. Le mot IMG prépare l'imprimante à 242 impressions graphiques. Aucune variable n'est utilisée, les informations transitent toutes par la pile.

Pour d'autres machines, la longueur des boucles doit être modifiée en fonction de la largeur et de la hauteur en pixels de l'écran ; le mot LEMIT doit être remplacé par celui qui permet de sortir une information sur l'imprimante, et le mot POINT par celui qui permet de savoir si un pixel est d'une couleur différente de celle du fond.

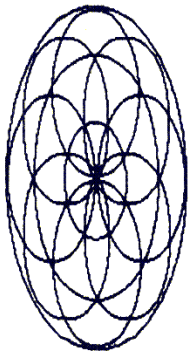
Ces programmes peuvent aussi être adaptés à une autre imprimante. Dans ce cas, il faut remplacer les codes de contrôle pour l'impression graphique (voir les indications des manuels), et si l'imprimante possède une matrice de sept aiguilles il faut modifier le pas de la boucle externe (il passera de 8 à 7).

Michel BROCHAND

Programme Copiec

```
: 2PN DUP IF 1 SWAP 0 DO 2* LOOP ELSE
  DROP 1 THEN ; : LEM LEMIT ;
: IMG 27 LEMIT 71 LEM 0 LEM 242 LEM ;
: COPIEC 27 LEM 48 LEM 0 0 227 DO IMG
 I 242  0 DO I 8 0 DO DUP 3 PICK I -
 POINT 0= NOT IF ROT I 2PN + ROT ROT
 THEN LOOP  DROP SWAP LEM  0 SWAP
 LOOP  10 LEM   DROP  -8 +LOOP ;
```

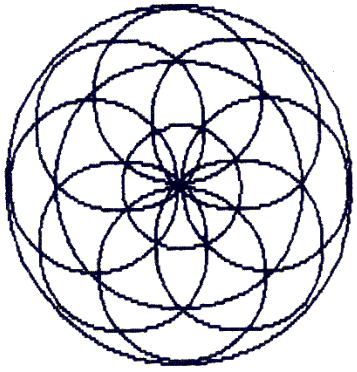
Exemple d'exécution de Copiec



Programme Copiw

```
: PRE 10 227 160 210 WINDOW ;
: IMPC 27 LEM 71 LEM 1 LEM 64 LEM ;
: COPIW 27 LEM  48 LEM 0 0 227 DO IMPC
 I 171  9 DO I 8 0 DO DUP 3 PICK I -
 POINT 0= NOT IF ROT I 2PN + ROT ROT
 THEN LOOP DROP SWAP DUP  LEM  LEM
 0 SWAP LOOP  10 LEM DROP -8 +LOOP ;
```

Exemple d'exécution de Copiw



GESTION DE BASE DE DONNÉES

ÉCRIT en Basic sur un Epson PX-8, le programme Pêle-Mêle se présente comme une base de données qui gère de petits fichiers. De la gestion des dîners pour la maîtresse de maison au carnet de rendez-vous des dentistes, ou même au dénouement d'une affaire policière, ce programme devrait résoudre bien des problèmes.

Qui a mangé de la charlotte aux fraises ? Et, plus généralement, qui a déjà mangé quoi ? Ce petit problème, surtout préoccupant pour les maîtresses de maison qui reçoivent beaucoup, sera résolu, comme bien d'autres, par Pêle-Mêle, un embryon de base de données qui gère autant de petits fichiers que vous voulez. Chaque fichier peut contenir jusqu'à 240 données, de 12 caractères au plus, corrélées entre elles par un maximum de 255 « liens », assemblant jusqu'à quatre données chacun.

Une pression de touche suffit pour choisir un fichier. La liste des données, sur 40 lignes de 6 colonnes, restera disponible en permanence : elle apparaît ou disparaît d'une pression de touche et défile à volonté à l'aide des curseurs verticaux sans délai d'affichage.

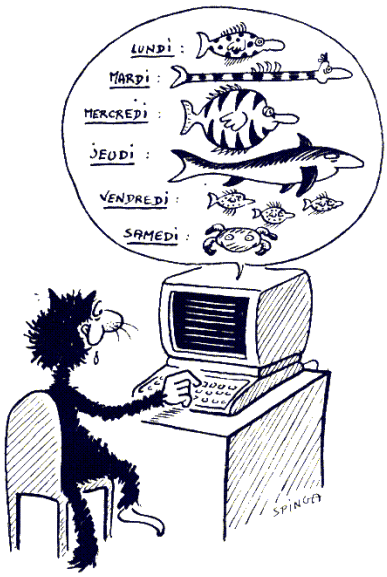
La touche RETURN ramène sur l'autre écran virtuel où un MENU (ligne 24, en DATA ligne 64) offre cinq options : l'option 1 ramène à la liste des données (ligne 22), l'option 2 donne les « liens » d'une donnée (ligne 26), l'option 3 permet la création (ligne 40) et l'option 4 la suppression d'un lien (ligne 48). Quant à RETURN, il réenregistre s'il y a lieu les fichiers sur disque RAM (spécifique au PX-8) ou cassette et met fin au programme (ligne 56).

En appelant le nom d'une donnée (option 2), on voit apparaître toutes celles qui ont un lien avec elle. Tous les détails subalternes sont réglés automatiquement : un même nom de donnée ne s'enregistre qu'une fois dans la liste, la suppression d'un lien efface toutes les données sans autre lien que celui-là, les trous ainsi créés ne se voient pas à l'affichage de la liste, et ils se rebouchent tout seuls au fur et à mesure de l'introduction de nouvelles données, etc.

L'occupation de mémoire est des plus raisonnables : 1 300 octets environ pour le programme et 2 à 4 Ko par fichier, dont 1 à 3 Ko en séquentiel, éventuellement sur cassette, et 1 Ko seulement en sélectif. Mais l'avantage essentiel de ce petit programme de gestion de liens est sa polyvalence qui lui permet d'entrer en compétition avec bien des programmes spécialisés.

Le menu et le dentiste

Reprenons le cas de la maîtresse de maison modèle qui offre par exemple deux plats et un dessert mémorables à six couples d'invités, au plus, chaque fois. Chaque dîner va être représenté par trois liens, l'un entre le menu et la date, qui sert ici de pivot, les autres entre la date et les noms de trois des ménages d'invités.



Même si elle dispose d'une cinquantaine de bonnes recettes et d'une centaine de ménages d'amis, elle pourra garder sous la main, et en 4 Ko seulement, tous les menus et toutes les listes d'invités des 85 derniers dîners qu'elle aura offerts.

Autre exemple, l'agenda d'un dentiste, devant contenir jusqu'à 250 rendez-vous. Ceux-ci s'étalent sur près de deux mois, soit une quarantaine de jours ouvrables, et ils correspondent, de quart d'heure en quart d'heure, à une trentaine de moments de la journée. D'autre part, ils n'auront pas été donnés à plus de 170 clients différents : 40 + 30 + 170 = 240, on peut donc y aller. Voyons la chose de plus près.

Un patient s'assied sur le fauteuil. Le dentiste tape un 4 (suppression de lien), puis le nom du client. Tous les rendez-vous de celui-ci s'affichent, ce qui permet de les lui confirmer. Et tapant le numéro de celui d'aujourd'hui, on l'annule avec retour au menu.

Une date est ensuite convenue pour un nouveau rendez-vous. Un 2, suivi de cette date, affiche tous les rendez-vous déjà fixés pour ce jour-là, ce qui permet de convenir d'une heure, et on aurait pu aussi bien choisir la date en fonction de l'heure. Enfin un 3, le nom du client, la date et l'heure convenues suffisent à enregistrer le nouveau rendez-vous.

Dès qu'un programme fait appel au système d'exploitation d'un micro, il est difficile de le transporter sur un autre. Bon prétexte pour ne pas nous priver ici

Pêle-Mêle
Programme pour Epson PX-8
Auteur Pierre Barnouin
Copyright *LIST* et l'auteur

```
10 DEFSTR A-H:DIM A(240):WIDTH 80,8,40:ON ERROR GOTO 60
12 READ E,D:PRINT E:E="A:L"+INPUT$(1)+".DAT":F="A:"+MID$(E,4)
14 SCREEN,1,0:OPEN"R",#1,E,255:FIELD#1,255 AS H
16 OPEN"I",#2,F:INPUT#2,G:WHILE EOF(2)=0:L=L+1:INPUT#2,A(L)
18 IF A(L)>""THEN PRINT TAB(13*K+1);A(L);:K=(K+1)MOD 6
20 WEND:CLOSE#2:FOR I=1 TO 4:GET#1,I:B(I)=LEFT$(H,VAL(G)):NEXT
22 ON ERROR GOTO 0:SCREEN,1:LINE INPUT;A:SCREEN,0
24 CLS:PRINT D:ON VAL(INPUT$(1)) GOTO 22,26,40,48:ON Z GOTO 56:END
26 GOSUB 28:LINE INPUT;A:GOTO 24
28 GOSUB 38:FOR K=1 TO L:J=J-K*(G=A(K)):NEXT:K=0
30 IF J=0 THEN PRINT"Elément inconnu":LINE INPUT;A:GOTO 24
32 FOR I=1 TO 4:K=INSTR(K+1,B(I),CHR$(J)):ON SGN(K)GOSUB 34:NEXT:RETURN
34 PRINT K;:FOR S=1 TO 4:T=ASC(MID$(B(S),K,1)):IF T-J THEN PRINT A(T)+"  ";
36 NEXT:I=I-1:PRINT.RETURN
38 J=0:INPUT"Nom d'élément";G:G=LEFT$(G,12):RETURN
40 Z=1:FOR I=1 TO 4:GOSUB 38:IF G=""THEN 46
42 FOR K=1 TO L:IF A(K)=""THEN J=K
44 WHILE A(K)=G:J=K:K=L+1:WEND:NEXT:IF J=0 THEN L=L+1:J=L
46 A(J)=G:B(I)=B(I)+CHR$(J):NEXT:GOTO 24
48 Z=1:GOSUB 28:INPUT"Numéro du lien à supprimer";K:FOR I=1 TO 4
50 C(I)=MID$(B(I),K,1):B(I)=LEFT$(B(I),K-1)+MID$(B(I),K+1):NEXT:FOR I=1 TO 4
52 K=0:FOR J=1 TO 4:K=K+INSTR(B(J),C(I)):NEXT:IF K=0 THEN A(ASC(C(I)))=""
54 NEXT:WHILE A(L)="":L=L-1:WEND:GOTO 24
56 A(0)=STR$(LEN(B(1))):FOR I=1 TO 4:LSET H=B(I):PUT#1,I:NEXT
58 KILL F:OPEN"O",#2,F:FOR I=0 TO L:PRINT#2,A(I):NEXT
60 IF ERR=53 THEN OPEN"O",#2,F:CLOSE#2:RESUME 22
62 DATA 1 CARNET        2 AGENDA        3 DINERS        4 FOURRE-TOUT
64 DATA LISTE   1 des éléments  2 liens d'un élément  RETURN = retour au menu
        LIENS   3 Création      4 Suppression         RETURN = Sauvegarde et FIN
```

GESTION DE BASE DE DONNÉES

des avantages spécifiques du PX-8 page-écran virtuelle de 40 lignes de 80 caractères, disque RAM non volatil et instructions d'un Basic poussé.

Il faudra de toute façon repenser le problème en fonction de l'architecture et du Basic du micro dont on dispose. Essayons de faciliter cette tâche en « soulevant le capot » et en détaillant quelques explications.

Tout le mécanisme repose sur les chaînes B(1) à B(4), contenant chacune jusqu'à 255 octets, dont chaque code ASCII représente l'indice d'une donnée stockée dans le tableau A(i) (A et B sont bien des chaînes, voir le DEFSTR en tête de programme). Chaque groupe de quatre octets de même rang dans les chaînes B(1) à B(4) matérialise un « lien » entre les données correspondant à ces octets. Déjà un premier barrage : les chaînes B(i) ne pourront généralement pas être stockées en séquentiel sans risque d'erreur. Elles peuvent être tronquées si elles contiennent le caractère « retour chariot », etc.

Et si l'on n'a pas de PX-8 ?

C'est une instruction INSTR à trois opérandes, INSTR(N,A$,B$), cherchant B$ à partir du énième caractère de A$, qui effectue quasi instantanément les recherches dans ces chaînes. On pourrait recourir à des boucles, mais ce serait un peu plus lent, et plus difficile.

Le réaffichage d'une liste de 255 données est forcément assez lent. On l'évite sur le PX-8 en se limitant aux 240 qu'on peut garder en permanence sur les 40 lignes de l'écran virtuel. La solution à adopter sur d'autres matériels conservera sans doute un réaffichage.

D'autres détails pourront être utiles un jour ou l'autre :

- ON ERROR GOTO, lignes 10 et 60, permet de créer ici, très économiquement et sans même s'en apercevoir, un fichier qui n'existerait pas encore ;
- ON ERROR GOTO 0, ligne 22, rétablit la procédure normale ;
- le nombre et les noms des fichiers seront fixés à votre gré en modifiant le DATA de la ligne 62. Le programme, lui, rebaptise tout bêtement 1, 2, 3, etc. les fichiers de données et L1, L2, L3, les fichiers de liens correspondants ;
- un « A » changé en « H », ligne 12, suffit pour mettre sur cassette les fichiers séquentiels de données.

Vous trouverez la façon :

- d'éviter l'affichage des « trous », ligne 18 ;
- de les reboucher par de nouvelles données, ligne 42 ;
- sauf si celles-ci figurent déjà dans la liste, ligne 44 ;
- de supprimer les données qui n'ont plus de lien, ligne 52 ;
- et aussi les trous situés en fin de liste, ligne 54.

Il sera alors possible de tester ce que vaut *Pêle-Mêle* comme débrouilleur d'embrouilles : confiez-lui les éléments d'une intrigue policière et c'est bien le diable si, de rapprochement en rapprochement, il ne fournit pas de quoi confondre l'assassin plus vite que le génial détective des romans.

Pierre BARNOUIN

D'un Basic à l'autre

Certaines instructions du PX-8 n'existent pas sur d'autres machines. Par exemple, pour remplacer une ligne comme :

```
100 ON VAL(INPUT$(1)) GOTO 200,
    300
```

on écrira :

```
100 A$=INKEY$:IF  A$=""  THEN
    100
110 IF A$="1" THEN 200
120 IF A$="2" THEN 300...
```

De la même façon, l'affichage de toutes les positions successives de B$ dans A$, qui s'écrit en une ligne sur PX-8 :

```
100 K=INSTR(K+1,A$,B$):IF  K
    THEN PRINT K:GOTO 100
```

peut s'écrire :

```
100 FOR I=1 TO LEN(A$)
110 IF B$=MID$(A$,I,LEN(B$))
    THEN PRINT I
120 NEXT I
```

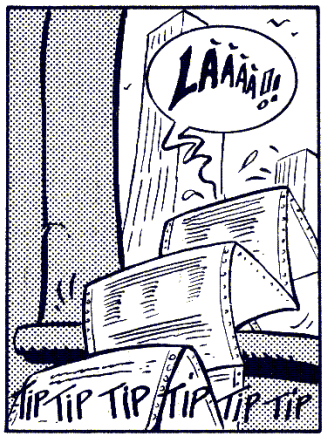
Enfin, la boucle WHILE...WEND, soit :

```
100 WHILE <condition> : <instructions> : WEND
```

peut être remplacée par :

```
100 IF <condition> THEN <instructions> : GOTO 100
```

Les < instructions > entre WHILE et WEND (ou entre THEN et GOTO) doivent pouvoir modifier la < condition >, sans quoi on entre dans une boucle sans fin. Si ces < instructions > comprennent l'incrémentation d'une variable, on remplacera aisément WHILE...WEND par une boucle FOR...NEXT.



ORIGINES

LA DÉESSE CANON, LA POMME ET LE GUÉPARD

Il est très rare qu'un mot, ou qu'un nom, soit forgé de toutes pièces ou choisi au hasard. Presque toujours, il possède une origine, certaines raisons qui l'ont fait préférer à d'autres. L'informatique n'échappe pas à cette règle et les ordinateurs ne s'appellent pas n'importe comment. Voici quelques exemples variés d'étymologies.

AMSTRAD

Ce nom est l'abréviation de Alan Michael Sugar TRADing corporation. Alan Michael n'est autre que le fondateur de la société.

Jusqu'à présent, tous les ordinateurs de cette firme sont appelés CPC, pour Colour Personnal Computer. Le premier chiffre 4 du CPC 464 semble signifier qu'un lecteur de cassettes est intégré, 64 indique que la capacité de mémoire vive est de 64 Ko. Le premier 6 du CPC 664 correspondrait au lecteur de disquettes intégré. Le nouvel Amstrad CPC 6128, confirme d'ailleurs cette supposition : le CPC 6128 est un Colour Personnal Computer avec lecteur de disquettes intégré et 128 Ko de mémoire vive.

APPLE

A l'origine, Jobs, Wozniak et Markkula, les trois fondateurs de la firme, ne savaient pas quel nom donner à leur compagnie le jour où ils allèrent la faire enregistrer. Comme il mangeait une pomme, Jobs suggèra le nom d'Apple (*pomme*, en français), justement. Ce nom avait pour lui « d'être séduisant, de n'être ni menaçant ni anonyme comme XYZ, et d'être employé à l'école pour enseigner aux enfants la pre-

OFFRE EXCEPTIONNELLE

LES 11 PREMIERS NUMEROS DE LIST

LE JOURNAL DES MAÎTRES DE PROGRAMMATION

109F

RETOURNEZ LE BON CI-CONTRE

BON DE COMMANDE

(à retourner à LIST - Service Numéros - 5, place du Colonel-Fabien - 75491 Paris Cedex 10).

Nom _____ Prénom _____

Adresse _____

_____

Code Postal |_|_|_|_|_| Ville _____

Je désire recevoir la collection complète des 11 premiers numéros de **LIST** au prix de **109 F.** Je joins mon règlement indispensable à l'ordre de **LIST.**

(Offre valable pour la France métropolitaine).

LI 12

mière lettre de l'alphabet ».

Le nom de LISA est l'abréviation de Local Integrated Software Architecture. Le Macintosh est une variété de pommes. Ce terme était utilisé comme nom de code pendant la période de conception du matériel, mais sonnant bien et l'habitude aidant, il fut conservé.

CANON

En 1933, une douzaine d'hommes créèrent une entreprise qui devait fabriquer, deux ans plus tard, le premier appareil de photos japonais. Cet appareil fut baptisé Kwanon, du nom de la déesse bouddhiste de la merci, de la grâce accordée.

Enfin, en 1947, la société prendra le nom de Canon Camera Co Ltd afin d'adapter son image de marque au caractère international de ses activités.

Côté informatique, le X-07 n'est autre que le code donné lors de la réalisation de prototypes. Ce n'est qu'au moment de son lancement que l'on décida de conserver ce nom, le X symbolisant les facultés scientifiques du produit.

CASIO

La société Casio a été fondée par les frères Kashio, l'orthographe de leur nom ayant été simplifiée pour obtenir la raison sociale de la firme. L'origine des noms des différents modèles n'est pas toujours évidente, mais on connaît celle des principaux :

- SL est mis pour les machines solaires (SL = Solar) ;
- H correspond généralement à des machines dont la taille permet de les tenir dans la main (H = Handy) ;
- les modèles dont le nom commence par J sont des machines de la taille au-dessus (J = Just) ;
- tous les modèles scientifiques commencent par FX ;
- les modèles de poche non scientifiques et fonctionnant en langage Basic commencent par PB (PB = Pocket Basic) ;
- les calculatrices imprimantes de bureau commencent par la lettre D (D = Desk).

GUÉPARD

La société qui a conçu l'ordinateur Guépard s'appelle HBN Electronic. Son nom est formé par certaines lettres du patronyme de son fondateur, Jean-Claude HouBroN.

Le nom du micro-ordinateur Guépard a une origine liée à la définition qu'en donne le Petit Larousse : « Animal réputé pour son extrême rapidité et sa domestication facile ».

SHARP

C'est en 1915 que Tokuji Hayakawa inventa au Japon le premier stylo mine à avance mécanique, l'Ever Sharp Pencil (littéralement, *le stylo toujours pointu*). Ce premier produit donna son nom à la société Sharp qui diversifia ensuite ses productions.

Quant aux noms des différents ordinateurs, ils correspondent généralement aux initiales de termes techniques :

- PC est fait des initiales de Pocket Computer ;
- EL est mis pour Elsimate (« Compagnon LSI », *Large Scale Integrated*) ;
- ER pour Electronic Cash Register ;
- MZ pour Micro Zilog (gamme de micro-ordinateurs) ;
- SPC : Super Portable Computer ;
- SF : Sharp Fax (Fac simile).

TANDY

Charles Tandy, fondateur du groupe Tandy, racheta la société d'électronique Radio Shack, il y a une vingtaine d'années, d'où Tandy Radio Shack, ou TRS. Le premier ordinateur conçu par la société étant construit autour d'un microprocesseur Z80, il fut baptisé TRS-80.

LAISSEZ-VOUS COPIER...

Duplication de disquettes
3"/3½"/5¼"
Tous formatages

Duplication express 24H Soft Assistance

KBI
109, bureaux de la Colline de St Cloud
92210 ST CLOUD
Actuellement (1) 602.40.00
A partir du 25 Oct. (1) 46.02.40.00

SOLUTIONS DES JEUX DE LA PAGE 22

LES problèmes proposés à la page 22 appellent des solutions. Voici celles que nous proposons.

39

Messages d'erreur

1-B
2-C
3-H
4-D
5-A
6-J
7-G
8-F
9-E
10-I

40

Optimisation

Les deux programmes proposés sont équivalents lorsque N1 est inférieur ou égal à N2. En revanche, lorsque N1 est supérieur à N2, la boucle de la ligne 10 n'est pas exécutée (sauf pour quelques langages, autres que le Basic). Dans ce cas, la première version n'effectue pas l'initialisation du tableau, alors que la seconde le fait systématiquement. Ce qui peut trahir l'intention du programmeur. De plus, si la seconde version est plus rapide lorsque N1 est inférieur ou égal à N2, elle est plus lente dans le cas contraire.

41

Trouver des formules

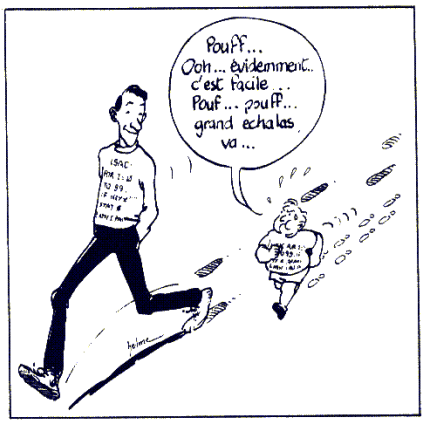
• Première formule : (B – A) * (I MOD 2) + A

L'opérateur MOD retourne le reste de division entière. S'il est absent, l'expression devient : (B – A)*(I – 2*ENT(I/2)) + A où ENT(x) est la partie entière de X.

• Deuxième formule : ((B – A) – (B – A) * $(-1)^I$) / 2

• Troisième formule : (B – A) * ABS(SIN(90*I)) + A, si l'ordinateur travaille en degrés. S'il travaille en radians (avec PI = 3.141592654), l'expression doit être remplacée par : (B – A) * ABS(SIN((PI/2)*I)) + A.

• Quatrième formule : (A – B) * ABS(COS(90*I)) + B, si l'ordinateur travaille en degrés. De même que précédemment, s'il travaille en radians, on a : (A – B)*ABS(COS((PI/2)*I)) + B.



Optimisation : deux programmes, deux vitesses

42

Les deux plus grands

Les deux programmes proposés donnent un résultat faux si la valeur du plus grand élément apparaît plus d'une fois dans le tableau : ils donnent la même valeur aux deux éléments recherchés. Les corrections à apporter sont les suivantes :

• dans le premier programme, la ligne 70 doit être remplacée par :
70 IF T(M2) < T(I) AND M1 < > I THEN M2 = I

• dans le second programme, c'est la ligne 40 qui doit être remplacée par :
40 IF T(M1) < T(I) THEN M1 = I: GOTO 60

43

Langages

1. APL : A Programming Language (un langage de programmation, "*un* de plus", en opposition avec PL/1)

2. Basic : Beginner's All Purpose Symbolic Instruction Code

3. Cobol : COmmon Business Oriented Language

4. Fortran : FORmula TRANslation

5. GAP : Générateur Automatique de Programmes (nom français du RPG, Report Program Generator)

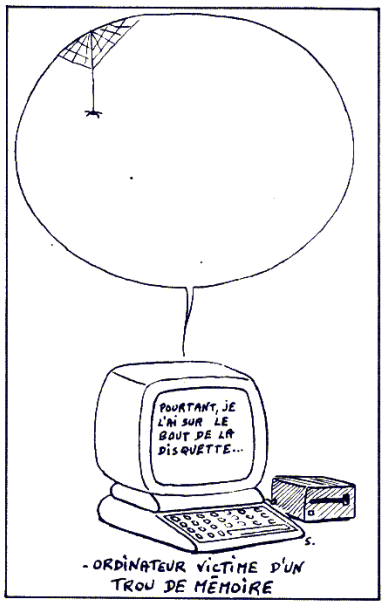
6. Lisp : LISt Processor

7. LSE : Langage Symbolique d'Enseignement

8. Pascal : il ne s'agit pas d'une abréviation, mais du nom de Blaise Pascal.

9. PL/1 : Programming Language 1 (*LE* langage de programmation numéro 1, à opposer à APL)

10. Prolog : PROgramming in LOGic



44

Tous les sens

Le langage L considère l'instruction A = B = 0 comme une affectation en chaîne exécutée à partir de la droite. Dans un premier temps, la valeur 0 est mémorisée dans B, puis dans A. C'est pourquoi cette dernière contient 0 avec le langage L. Ce pourrait être, par exemple, certains rares Basic.

Le langage M considère cette instruction comme une affectation de variable logique. Dans un premier temps, c'est l'expression (B = 0) qui est évaluée. Si elle est vraie, la valeur 1 est stockée dans A (selon les ordinateurs, ce peut être une autre valeur). Le langage Pascal fonctionne de cette façon, à la différence que l'instruction s'écrit : "A: = B = 0".

Le langage N fonctionne comme le langage L mais en partant de la gauche. En partant d'une valeur de B égale à 2, la première partie de l'instruction (A = B) impute la valeur 2 à A. La seconde partie de cette instruction (B = 0) remet la variable B à zéro. Il existe toutefois peu de langages fonctionnant de cette façon.

45

Racine numérique

La plus courte fonction capable de donner la racine numérique d'un nombre est :

RN(N) = N MOD 9

où l'opérateur MOD (modulo) retourne le reste de la division entière de N par 9. En Basic, cette fonction peut s'écrire :

DEFFN RN(N) = N − 9*INT(N/9)

En Pascal, elle se déclare ainsi :

function racinum (n : integer) : integer;
begin
  racinum: = n *mod* 9
end;

Cette formule qui donne le reste de la division par 9, est utilisée dans ce que l'on appelait la "preuve par neuf" à l'école primaire.

46

Rapports ambigus

Le libellé de l'erreur est de la forme : "floating point overflow", c'est-à-dire "dépassement de capacité en réel". Des valeurs suffisamment grandes de A, B, C ou D déclenchent l'erreur. C'est le cas, par exemple, avec un ordinateur dont la capacité pour les nombres réels atteint 9.9E99 et des valeurs de A, B, C ou D comme : A = 1.0E51, B = 1.0E52, C = 1.0E48, D = 1.0E49.

Les calculs de A/B et C/D s'effectuent sans problème, puisque dans les deux cas on obtient 1.0E − 1. Mais, A*D et C*B donnent 1.0E100 qui dépasse la capacité de l'ordinateur. D'où le message d'erreur.

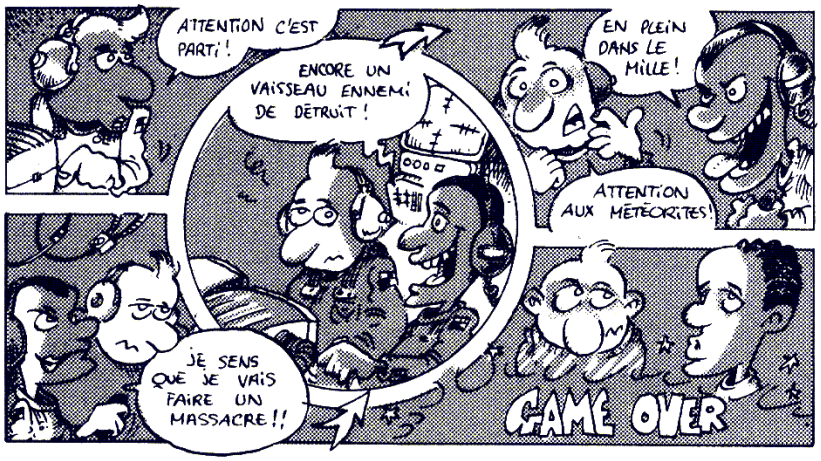
47

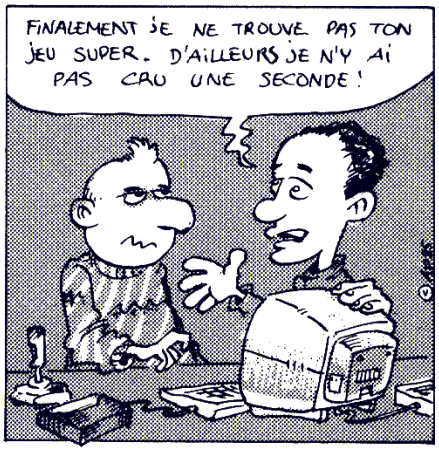
Les surprises du calendrier

En prenant OCT comme abréviation de "octal" et DEC pour "décimal", on a :

31 OCT = $31_8 = (3*8 + 1)_{10} = 25_{10}$ = 25 DEC

(D'après une nouvelle d'Isaac Asimov, "Un cas étrange de fraude fiscale" — *A curious case of income tax.*)





Imprimerie SIMA, rue des Epinettes, ZI de Torcy. Printed in France. Directeur de la publication : Jean-Luc Verhoye.
Dépôt légal septembre 1985. Diffusion NMPP. Numéro de commission paritaire : 63617.

CASIO

INITIATION BASIC
FX 720 P

FX 720 P : MINI-ORDINATEUR DE POCHE